Quantstamp

Mezo Portal

Executive Summary

This audit report was prepared by Quantstamp, the leader in blockchain security.

Type	Bitcoin Bridge
Timeline	2024-04-29 through 2024-05-03
Language	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	RFC: OrangeKit Bitcoin Account Metaprotocol
Source Code	• thesis/mezo-portal #0000ff5 • https://github.com/keep-network/tbtc-v2 #9e047d1 • https://github.com/thesis/orangekit #44355ad
Auditors	• Cameron Biniamow Auditing Engineer • Shih-Hung Wang Auditing Engineer • Rabib Islam Auditing Engineer

Documentation quality	High
Test quality	High
Total Findings	10 Fixed: 3 Acknowledged: 7
High severity findings	0
Medium severity findings	1 Fixed: 1
Low severity findings	2 Acknowledged: 2
Undetermined severity findings	0
Informational findings	7 Fixed: 2 Acknowledged: 5

Summary of Findings

Mezo is a project focused on developing an "economic layer" for Bitcoin. The current audit report is concerning a Points Portal where users can deposit BTC to earn points.

In order to participate in Mezo Portal, users are to deposit BTC on the Bitcoin network to an address determined by a script and its particular inputs. Following that, a process can be initiated that will result in TBTC being deposited into the `Portal` contract for a user-determined, protocol-constrained, pre-specified lock period.

During this audit, we found an issue which results in a loss of security for the ECDSA being used to validate Bitcoin signed messages that enable the use of an admin function.

Overall, the code quality was quite good. We do recommend, however, updating the test suite in order to ensure that all tests are passing.

Update: The issues have been addressed.

ID	DESCRIPTION	SEVERITY	STATUS
MEZO-1	Potential Signature Forgery Due to Lack of Validation on Public Keys	● Medium	Fixed
MEZO-2	Late Lock Period Validation May Lead to Stuck Funds	● Low	Acknowledged
MEZO-3	Incompatible with Deflationary and Fee-on-Transfer Tokens	● Low	Acknowledged
MEZO-4	`PUSH0` Remains Unsupported on some Blockchains	● Informational	Acknowledged

ID	DESCRIPTION	SEVERITY	STATUS
MEZO-5	Inconsistent Event Emission	● Informational ⓘ	Acknowledged
MEZO-6	Gas Savings	● Informational ⓘ	Acknowledged
MEZO-7	Unmasked Result of Create2 Address Calculation	● Informational ⓘ	Fixed
MEZO-8	ECDSA Signature Malleability	● Informational ⓘ	Fixed
MEZO-9	Potential Incompatibility with ERC-4337	● Informational ⓘ	Acknowledged
MEZO-10	Privileged Roles and Ownership	● Informational ⓘ	Acknowledged

Assessment Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

Disclaimer
Only features that are contained within the repositories at the commit hashes specified on the front page of the report are within the scope of the audit and fix review. All features added in future revisions of the code are excluded from consideration in this report.

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

1. Code review that includes the following
   1. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
   2. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   3. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
   1. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   2. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Scope

Files Included

Scope for `thesis/mezo-portal`

- `solidity/contracts/BitcoinDepositor.sol`
- `solidity/contracts/Portal.sol`

Scope for `keep-network/tbtc-v2`

- `solidity/contracts/integrator/AbstractTBTCDepositor.sol`

Scope for `thesis/orangekit`

- `solidity/contracts/*`

Findings

MEZO-1
Potential Signature Forgery Due to Lack of Validation on Public Keys

• Medium ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `51985d2`.

A function `isOnCurve()` is now being used to determine whether a point is on the curve. Although the fix appears to be sufficient, an extra level of assurance may be provided by validating that `x` and `y` are both below `SECP256K1_P`

File(s) affected: `BitcoinSafeOwner.sol`

Description: The `validateCompressedP2PKH()` function defined in `BitcoinSafeOwner` is susceptible to a hash collision attack (specifically, a birthday attack), where an adversary may try to brute-force different `y` and `s` values to forge a valid signature if the `x` value for calculating the `truncatedBitcoinAddress` is known. The collision attack would reduce the bits of security level from 160 to 81 bits, making the attack practical for well-capitalized attackers. The `validateP2SH_P2WPKH()` and `validateP2WPKH()` functions have the same issue.

Ultimately, this may lead to a situation where the singleton is upgraded to a malicious contract.

Exploit Scenario: We demonstrate the details of the attack as follows. Given a known `truncatedBitcoinAddress`, derived by a compressed public key `(x, y)`, our goal is to find some `y'`, `v`, `r`, `s` such that `ecrecover(signedMessage, v, r, s) == publicKeyToEthereumAddress(x, y')` for our chosen `signedMessage`. If so, we successfully forge a signature for `signedMessage`.

First, we randomly select `2^80` `y'` values whose last bit is the same as `y`. We calculate `publicKeyToEthereumAddress(x, y')` for each `y'` and collect the results (which are random addresses) to a set, `A`. Since `y` and `y'` have the same last bit, the derived `truncatedBitcoinAddress` remains the same.

Next, we set `v = 27` and `r` to an arbitrary constant, e.g., `bytes32(1)`. We randomly select `2^80` `s` values within the range of `[1, n - 1]`, where `n` is the order of the secp256k1 group. We calculate `ecrecover(signedMessage, v, r, s)` for each `s` and collect the results (also random addresses) to another set, `B`. Note that for any `s` in the specified range, `ecrecover()` should be able to recover a signer successfully with a negligible probability of failure, following the public key recovery process.

We compare the two sets, `A` and `B`. If any address in set `A` is also in `B`, we successfully achieve our goal. Given that the total number of addresses is `2^160`, since we have `2^80` uniformly sampled addresses in both sets `A` and `B`, we may find a collision with a reasonable probability, which is about `1 - 1/e = 0.63`.

Recommendation: When validating signatures for addresses derived from a compressed public key, consider adding a check to ensure that the provided `(x, y)` is a valid point on the secp256k1 curve. This would ensure the uniqueness of the `y` value for a given `truncatedBitcoinAddress`. Therefore, the above attack technique would become invalid.

MEZO-2 Late Lock Period Validation May Lead to Stuck Funds

• Low ⓘ Acknowledged

Update
Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
In the BTC deposit flow, the user needs to assemble the P2WSH address and the lock period is a part of
the script. This is quite a complicated action and users are not doing it manually. This happens in the
dApp and is implemented in the tBTC SDK: https://github.com/keep-network/tbtc-v2/tree/main/typescript.
In the context of the audit, we assume this code works correctly because if not, much worse things can
happen like, for example, sending tBTC to some arbitrary address without the P2WSH deposit script
encoded at all. If we assume the code of tBTC SDK works correctly, there is still one scenario when
MEZO-2 can happen: it's when the allowed lock time range changed between the time P2WSH script was
assembled and the deposit was revealed and finalized. But for this to happen, the governance must
execute an update that will cause this problem. I assume that if the need for such an upgrade arises,
the governance will execute it in a responsible way, like updating the allowed range in the dApp before
changing it on the contract side and making sure there are no deposits in the queue that would violate
the rules.
```

File(s) affected: `BitcoinDepositor.sol`, `Portal.sol`

Description: In the flow for `BitcoinDepositor.finalizeDeposit()`, the function `Portal._calculateUnlockTime()` is called, and the validation at `Portal.sol#L461-467`, which checks whether `lockPeriod` is within a fixed range, may cause the transaction to revert. However, in the context of the protocol, this would be occurring after the deposit is revealed and after TBTC is already minted to the `BitcoinDepositor` contract. Moreover, there is no means implemented to recover this minted TBTC from the `BitcoinDepositor`.

Recommendation: Consider validating the `lockPeriod` in the flow for `BitcoinDepositor.initializeDeposit()` in order to avoid reversion of `finalizeDeposit()`.

MEZO-3
Incompatible with Deflationary and Fee-on-Transfer Tokens

• Low ⓘ Acknowledged

Update

Marked as "Acknowledged" by the client.
Addressed in: `347c04a`.
The client provided the following explanation:

```
We do not plan to work with deflationary or fee-on-transfer tokens. We added a warning about it and we
will make it a part of our governance action checklist when adding new supported tokens.
```

File(s) affected: `Portal.sol`

Description: If any tokens used are deflationary or have a fee-on-transfer and do not maintain a constant supply, fewer funds than expected could be transferred into the `Portal` contract when `_depositFor()` is executed. Therefore, the `deposits` mapping would hold an inflated `token` balance for the `depositOwner`. While the deposit would execute successfully, when the `depositOwner` attempts to withdraw their tokens, there could be an insufficient token balance in the `Portal` contract to support the withdrawal.

Recommendation: Avoid using deflationary or fee-on-transfer tokens in the `Portal` contract. If fee-on-transfer tokens are desired, check the token balance of the `Portal` contract before and after the transfer to obtain the actual amount of tokens transferred. Note that for deflationary tokens, it is difficult to track each user's deposit amount accurately, and additional logic would need to be added to the `Portal` contract to support these tokens.

MEZO-4
`PUSH0` Remains Unsupported on some Blockchains

• Informational ⓘ Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
The Mezo Portal smart contracts have the Solidity pragma fixed on version 0.8.24 and this version of
the compiler is used in hardhat.config.ts. We do not plan to deploy the Portal contract to L2s in the
near future but we will consider the PUSH0 limitation if we decide to do so.
  The OrangeKit smart contracts have the Solidity pragma fixed on version 0.8.25 and this version of
the compiler is used in hardhat.config.ts . PUSH0 could be a potential problem but we do not plan to
deploy OrangeKit contracts to L2s in the near future. We will consider this limitation if we decide to
do so.
```

Description: It should be noted that since Solidity version `0.8.20`, the `PUSH0` opcode is being used. However, some EVM blockchains may not support this opcode. Special care is advised given the potential changes to contract deployment code as well as the corresponding effect on contract addresses; if the same contract addresses are desired across all chains, the same compilation options should be used for every deployment.

Recommendation: Check whether the blockchains targeted for deployment support `PUSH0`. If it is desired to deploy on blockchains that do not implement `PUSH0`, it would be advised to compile with the `paris` EVM version.

MEZO-5 Inconsistent Event Emission

• Informational ⓘ Acknowledged

Update
Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
Acknowledged: https://github.com/thesis/mezo-portal/issues/811. Since the contracts are already
deployed and this issue is just informational, we will consider this change during the next potential
Portal contract upgrade.
```

File(s) affected: `Portal.sol`

Description: In the `initialize()` function, an array of `supportedTokens` are added to the contract. However, unlike in `addSupportedToken()`, the event `SupportedTokenAdded` is not emitted for each new token.

Recommendation: Consider emitting the `SupportedTokenAdded` event for each token added in the `initialize()` function for consistency.

MEZO-6 Gas Savings

● Informational ⓘ Acknowledged

Update
Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
Acknowledged: https://github.com/thesis/mezo-portal/issues/812. Since the contracts are already
deployed and this issue is just informational, we will consider this change during the next potential
BitcoinDepositor contract upgrade.
```

File(s) affected: `BitcoinDepositor.sol`, `AbstractTBTCDepositor.sol`

Description: Certain changes can be made to improve the gas efficiency of the contracts:
1. Use custom errors instead of `require()`.

Recommendation: Consider implementing the above recommendations.

MEZO-7 Unmasked Result of Create2 Address Calculation

● Informational ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `736822e`.

File(s) affected: `OrangeKitSafeFactory.sol`

Description: The `computeAddress()` function in the `OrangeKitSafeFactory` contract is borrowed from OpenZeppelin's `Create2` contract. Note that in a recent update of `Create2`, the returned `addr` value is masked to prevent dirty upper bits from being used later in assembly code blocks. See PR #4941 for more details.

Recommendation: We suggest following OpenZepplin's latest code by updating the corresponding line to:

```
addr := and(keccak256(start, 85), 0xffffffffffffffffffffffffffffffffffffffff)
```

MEZO-8 ECDSA Signature Malleability

● Informational ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `84dce96`.

It is now checked whether `s` is greater than the appropriate threshold value. If it is, the transaction reverts.

File(s) affected: `BitcoinSafeOwner.sol`

Description: The `BitcoinSafeOwner` uses `ecrecover()` to recover the signer from a given signature. Note that `ecrecover()` allows signature malleability, where two different `s` values can be combined with the same `r` value to produce two valid signatures.

The malleability of the `s` value does not cause an issue in the current use case of recovering the signer. Still, it is best practice to avoid signature malleability to enhance the code's robustness and prevent potential issues in future iterations.

Recommendation: Consider replacing `ecrecover()` with OpenZeppelin's `recover()`, which checks the `s` value to be within a specific range and raises an error if not.

MEZO-9 Potential Incompatibility with ERC-4337

● Informational ⓘ Acknowledged

Update
Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
The plan for achieving ERC-4337 compatibility has been outlined in
https://github.com/thesis/orangekit/pull/81.
```

The plan mentioned above consists of an RFC. However, we note that if implemented as stated, the plan may result in a roadblock due to rules in ERC-7562. We have followed up with the client.

File(s) affected: `BitcoinSafeOwner.sol`

Description: According to the given documentation "RFC: OrangeKit Bitcoin Account Metaprotocol", compatibility with ERC-4337 is one of the goals when designing the smart account.

Typically, when an ERC-4337 account validates a user operation, the signature validation logic is forwarded to the owner via the ERC-1271 flow if the owner is a contract, which is the approach implemented by the `Safe4337Module`. Therefore, the `isValidSignature()` function implemented by the owner contract has to comply with the ERC-4337 validation rules. Otherwise, the user operation could be rejected by bundlers.

Among the validation rules, the storage access rules restrict a non-entity contract to only access account-associated storage slots during the validation phase. If the owner contract is a proxy contract, a call to the owner would violate the storage access rule as the implementation slot is non-associated with the account. Since the owner is a `BitcoinSafeOwner` proxy, the call to `isValidSignature()` will be incompatible with the validation of ERC-4337 user operations.

A minor thing to note is that the `Safe` contract of version v1.4.1 does not support ERC-4337 by default. Instead, the `Safe4337Module` should be enabled at Safe deployment time afterward.

Recommendation: Consider how the smart account should support ERC-4337 and modify the contract to make the user operation validation flow comply with the ERC-4337 standard.

MEZO-10 Privileged Roles and Ownership

● Informational ⓘ Acknowledged

Update
Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
We inform our users about the custody model in our documentation: https://info.mezo.org/btc-custody-on-
mezo/deposit-custody but we will also improve it on the smart contract documentation level:
https://github.com/thesis/mezo-portal/issues/810.
```

Description: Smart contracts will often store specific addresses in order to accord them with special privileges, e.g. to make modifications to other important data.

The following are a list of functions that are only accessible to particular addresses:

1. `BitcoinSafeOwner`
   1. `truncatedBitcoinAddress`
   2. `upgradeSingleton()`
   3. `emergencyGovernance.emergencyUpgrader()`
   4. `emergencyUpgradeSingleton()`
2. `EmergencyGovernance`
   1. `owner`
   2. `disable()`
   3. `setEmergencyUpgrader()`
3. `OrangeKitSafeFactory`
   1. `owner`

   2. `upgradeSingleton()`
   3. `transferOwnership()`
4. `Portal`
   1. `owner`
   2. `addSupportedToken()`
   3. `setMinLockPeriod()`
   4. `setMaxLockPeriod()`

Note that some of the contracts are upgradeable, including `BitcoinDepositor`, `Portal`, `BitcoinSafeOwner`, and `OrangeKitSafeFactory`. Such contracts can have their implementations changed by the owner of the proxy.

Recommendation: This centralization of power needs to be made clear to the users, especially depending on the level of privilege the contract allows to the owner.

Definitions

- **High severity** – High-severity issues usually put a large number of users' sensitive information at risk, or are reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
- **Medium severity** – Medium-severity issues tend to put a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or are reasonably likely to lead to moderate financial impact.
- **Low severity** – The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
- **Informational** – The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
- **Undetermined** – The impact of the issue is uncertain.
- **Fixed** – Adjusted program implementation, requirements or constraints to eliminate the risk.
- **Mitigated** – Implemented actions to minimize the impact or likelihood of the risk.
- **Acknowledged** – The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).

Appendix

File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

Files

- `c40...82d ./mezo-contracts/EmergencyGovernance.sol`
- `fcc...8e3 ./mezo-contracts/LegacyERC1271.sol`
- `1c1...52b ./mezo-contracts/BitcoinDepositor.sol`
- `96b...d80 ./mezo-contracts/Proxy.sol`
- `b24...b8e ./mezo-contracts/Portal.sol`
- `6f4...313 ./mezo-contracts/ERC1271.sol`
- `bfd...7c2 ./mezo-contracts/BitcoinSafeOwner.sol`
- `691...c94 ./mezo-contracts/OrangeKitDeployer.sol`
- `711...bab ./mezo-contracts/OrangeKitSafeFactory.sol`
- `acf...89e ./mezo-contracts/AbstractTBTCDepositor.sol`

Tests

- `573...055 ./mezo-tests/keepnetwork-test/vault/TBTCVault.OptimisticMinting.test.ts`
- `1fc...4e7 ./mezo-tests/keepnetwork-test/vault/TBTCVault.Redemption.test.ts`
- `2ea...340 ./mezo-tests/keepnetwork-test/vault/TBTCVault.test.ts`
- `ef9...976 ./mezo-tests/keepnetwork-test/vault/DonationVault.test.ts`
- `355...79d ./mezo-tests/keepnetwork-test/helpers/contract-test-helpers.ts`
- `e4d...927 ./mezo-tests/keepnetwork-test/data/moving-funds.ts`
- `1c1...a3a ./mezo-tests/keepnetwork-test/data/deposit-sweep.ts`
- `e54...080 ./mezo-tests/keepnetwork-test/data/ecdsa.ts`

- `593...a19 ./mezo-tests/keepnetwork-test/data/redemption.ts`
- `366...f63 ./mezo-tests/keepnetwork-test/data/fraud.ts`
- `a83...f71 ./mezo-tests/keepnetwork-test/relay/LightRelayMaintainerProxy.test.ts`
- `bed...d84 ./mezo-tests/keepnetwork-test/relay/LightRelay.test.ts`
- `e1a...414 ./mezo-tests/keepnetwork-test/fixtures/bridge.ts`
- `403...771 ./mezo-tests/keepnetwork-test/fixtures/index.ts`
- `68a...cf0 ./mezo-tests/keepnetwork-test/l2/L2TBTC.test.ts`
- `14c...874 ./mezo-tests/keepnetwork-test/l2/L2WormholeGateway.test.ts`
- `9f0...99d ./mezo-tests/keepnetwork-test/integration/FullFlow.test.ts`
- `7ed...0a9 ./mezo-tests/keepnetwork-test/integration/WalletCreation.test.ts`
- `19d...ec7 ./mezo-tests/keepnetwork-test/integration/Slashing.test.ts`
- `414...acd ./mezo-tests/keepnetwork-test/integration/data/integration.ts`
- `88a...aa4 ./mezo-tests/keepnetwork-test/integration/data/bls.ts`
- `e8b...fbd ./mezo-tests/keepnetwork-test/integration/utils/random-beacon.ts`
- `3c7...150 ./mezo-tests/keepnetwork-test/integration/utils/ecdsa-wallet-registry.ts`
- `f0a...f0f ./mezo-tests/keepnetwork-test/integration/utils/gas.ts`
- `1ff...f7a ./mezo-tests/keepnetwork-test/integration/utils/staking.ts`
- `dc0...736 ./mezo-tests/keepnetwork-test/integration/utils/fake-random-beacon.ts`
- `b61...ad0 ./mezo-tests/keepnetwork-test/integration/utils/fixture.ts`
- `d0d...d72 ./mezo-tests/keepnetwork-test/maintainer/MaintainerProxy.test.ts`
- `a34...214 ./mezo-tests/keepnetwork-test/bridge/Bridge.Redemption.test.ts`
- `0e9...7c8 ./mezo-tests/keepnetwork-test/bridge/WalletProposalValidator.test.ts`
- `932...326 ./mezo-tests/keepnetwork-test/bridge/Bridge.Parameters.test.ts`
- `769...7b5 ./mezo-tests/keepnetwork-test/bridge/BitcoinTx.test.ts`
- `0e0...a51 ./mezo-tests/keepnetwork-test/bridge/Bridge.Wallets.test.ts`
- `84f...486 ./mezo-tests/keepnetwork-test/bridge/Bridge.Deposit.test.ts`
- `28f...e2a ./mezo-tests/keepnetwork-test/bridge/Bridge.Governance.test.ts`
- `c48...c29 ./mezo-tests/keepnetwork-test/bridge/VendingMachine.Upgrade.test.ts`
- `5ca...139 ./mezo-tests/keepnetwork-test/bridge/Bridge.MovingFunds.test.ts`
- `421...cb7 ./mezo-tests/keepnetwork-test/bridge/Heartbeat.test.ts`
- `604...ff9 ./mezo-tests/keepnetwork-test/bridge/Bridge.Vaults.test.ts`
- `5fc...4e8 ./mezo-tests/keepnetwork-test/bridge/Bridge.Frauds.test.ts`
- `e96...60e ./mezo-tests/keepnetwork-test/bridge/VendingMachineV3.test.ts`
- `26c...c27 ./mezo-tests/keepnetwork-test/bridge/VendingMachine.test.ts`
- `daf...52b ./mezo-tests/keepnetwork-test/bridge/Deployment.test.ts`
- `6d0...5b1 ./mezo-tests/keepnetwork-test/bridge/VendingMachineV2.test.ts`
- `027...ffa ./mezo-tests/keepnetwork-test/bridge/EcdsaLib.test.ts`
- `db4...95c ./mezo-tests/keepnetwork-test/bank/Bank.test.ts`
- `7df...bba ./mezo-tests/keepnetwork-test/integrator/AbstractTBTCDepositor.test.ts`
- `4c9...4ff ./mezo-tests/orangekit-test/OrangeKitSafeFactory.upgrades.test.ts`
- `405...0c4 ./mezo-tests/orangekit-test/BitcoinSafeOwner.test.ts`
- `8fe...2e8 ./mezo-tests/orangekit-test/OrangeKitDeployer.test.ts`
- `519...59f ./mezo-tests/orangekit-test/EmergencyGovernance.test.ts`
- `dd8...e3c ./mezo-tests/orangekit-test/SafeWithBitcoinOwner.test.ts`
- `9d9...32b ./mezo-tests/orangekit-test/BitcoinSafeOwner.upgrade.test.ts`
- `ccf...99e ./mezo-tests/orangekit-test/OrangeKitSafeFactory.test.ts`
- `48c...08e ./mezo-tests/orangekit-test/helpers/snapshot.ts`
- `018...d6e ./mezo-tests/orangekit-test/helpers/testBitcoinWallet.ts`
- `bac...aed ./mezo-tests/orangekit-test/helpers/bitcoinSafeOwner.test.ts`
- `b90...e99 ./mezo-tests/orangekit-test/helpers/bitcoinSafeOwner.ts`
- `2cc...9af ./mezo-tests/orangekit-test/fixtures/orangeKitFixture.ts`
- `c9b...0f0 ./mezo-tests/mezoportal-test/Portal.lock.test.ts`
- `1ac...60f ./mezo-tests/mezoportal-test/Portal.deposit.test.ts`
- `b39...cf5 ./mezo-tests/mezoportal-test/Portal.test.ts`
- `7bb...781 ./mezo-tests/mezoportal-test/BitcoinDepositor.test.ts`

- `75c...052 ./mezo-tests/mezoportal-test/Portal.receiveApproval.test.ts`
- `712...438 ./mezo-tests/mezoportal-test/Portal.upgrades.test.ts`
- `7ae...6d0 ./mezo-tests/mezoportal-test/Portal.withdraw.test.ts`
- `8e2...3ce ./mezo-tests/mezoportal-test/Portal.depositFor.test.ts`
- `e96...46b ./mezo-tests/mezoportal-test/fixtures/deployPortal.ts`
- `dc3...3d0 ./mezo-tests/mezoportal-test/integration/LockPeriod.test.ts`
- `dc8...a45 ./mezo-tests/mezoportal-test/integration/SupportedTokens.test.ts`
- `bc6...7d9 ./mezo-tests/mezoportal-test/integration/Depositing.test.ts`

Toolset

The notes below outline the setup and steps performed in the process of this audit.

Setup

Tool Setup:
- Slither v0.10.0

Steps taken to run the tools:
1. Install the Slither tool: `pip3 install slither-analyzer`
2. Run Slither from the project directory: `slither .`

Automated Analysis

Slither

No important issues were detected using Slither.

Test Suite Results

We were able to run all tests without failure.

```
Unit tests for 'keep-network/tbtc-v2'

  Bank
    PERMIT_TYPEHASH
      ✓ should be keccak256 of EIP2612 Permit message
    updateBridge
      when called by a third party
        ✓ should revert
      when called with 0-address bridge
        ✓ should revert
      when called by the governance
        ✓ should update the bridge
        ✓ should emit the BridgeUpdated event
    transferBalance
      when the recipient is the zero address
        ✓ should revert
      when the recipient is the bank address
        ✓ should revert
      when the spender has not enough balance
        ✓ should revert
      when the spender transfers part of their balance
        ✓ should transfer the requested amount
        ✓ should emit the BalanceTransferred event
      when the spender transfers part of their balance in two transactions
        ✓ should transfer the requested amount
      when the spender transfers their entire balance
        ✓ should transfer the entire balance
        ✓ should emit the BalanceTransferred event
      when the spender transfers 0 balance
        ✓ should transfer no balance
        ✓ should emit the BalanceTransferred event
```

```
approveBalanceAndCall
  when the spender is the zero address
    ✓ should revert
  when the spender callback reverted
    ✓ should revert
  when the spender had no approved balance before
    ✓ should approve the requested amount
    ✓ should emit the BalanceApproved event
    ✓ should call receiveBalanceApproval
  when the spender had an approved balance before
    ✓ should replace the previous allowance
    ✓ should call receiveBalanceApproval
approveBalance
  when the spender is the zero address
    ✓ should revert
  when the spender had no approved balance before
    when setting approval to non-zero amount
      ✓ should approve the requested amount
      ✓ should emit the BalanceApproved event
    when setting approval to zero
      ✓ should not change the zero approval
      ✓ should emit the BalanceApproved event
  when the spender had an approved balance before
    when setting approval back to zero
      ✓ should replace the previous allowance with zero
    when trying to overwrite with a non-zero value
      ✓ should revert
increaseBalanceAllowance
  when the spender is the zero address
    ✓ should revert
  when the spender had no approved balance before
    ✓ should approve the requested amount
    ✓ should emit the BalanceApproved event
  when the spender had an approved balance before
    ✓ should increase the previous allowance
  when the spender has a maximum allowance
    ✓ should revert
decreaseBalanceAllowance
  when the spender is the zero address
    ✓ should revert
  when the spender had no approved balance before
    ✓ should revert
  when the spender had an approved balance before
    ✓ should decrease the previous allowance
transferBalanceFrom
  when the recipient is the zero address
    ✓ should revert
  when the recipient is the bank address
    ✓ should revert
  when the spender has not enough balance approved
    ✓ should revert
  when the owner has not enough balance
    ✓ should revert
  when the spender transfers part of the approved balance
    ✓ should transfer the requested amount
    ✓ should emit the BalanceTransferred event
    ✓ should reduce the spender allowance
  when the spender transfers part of the approved balance in two transactions
    ✓ should transfer the requested amount
    ✓ should emit BalanceTransferred events
    ✓ should reduce the spender allowance
  when the spender transfers the entire approved balance
    ✓ should transfer the requested amount
    ✓ should reduce the spender allowance to zero
  when the spender transfers the entire balance
    ✓ should transfer the requested amount
    ✓ should reduce the spender allowance to zero
  when given the maximum allowance
    ✓ should not reduce the spender allowance
permit
  when permission expired
    ✓ should revert
```

```
        when permission has an invalid signature
          when owner does not match the permitting one
            ✓ should revert
          when spender does not match the signature
            ✓ should revert
          when permitted balance does not match the signature
            ✓ should revert
          when permitted deadline does not match the signature
            ✓ should revert
        when the spender is the zero address
          ✓ should revert
        when the spender had no permitted balance before
          ✓ should approve the requested amount
          ✓ should emit the BalanceApproved event
          ✓ should increment the nonce for the permitting owner
        when the spender had a permitted balance before
          ✓ should replace the previous approval
          ✓ should emit the BalanceApproved event
          ✓ should increment the nonce for the permitting owner
        when given never expiring permit
          ✓ should be accepted at any moment
      increaseBalance
        when called by a third party
          ✓ should revert
        when called by the bridge
          when increasing balance for the Bank
            ✓ should revert
          when called for a valid recipient
            ✓ should increase recipient's balance
            ✓ should emit the BalanceIncreased event
      increaseBalances
        when called by a third party
          ✓ should revert
        when called by the bridge
          when increasing balance for the bank
            ✓ should revert
          when there is more recipients than amounts
            ✓ should revert
          when there is more amounts than recipients
            ✓ should revert
          when called for a valid recipient
            ✓ should increase recipients' balances
            ✓ should emit BalanceIncreased events
      increaseBalanceAndCall
        when called by a third party
          ✓ should revert
        when called by the bridge
          ✓ should increase vault's balance
          ✓ should emit BalanceIncreased event
          ✓ should call the vault
          when depositors array has greater length than deposited amounts array
            ✓ should revert
          when deposited amounts array has greater length than depositors array
            ✓ should revert
      decreaseBalance
        ✓ should decrease caller's balance
        ✓ should emit the BalanceDecreased event
      DOMAIN_SEPARATOR
        ✓ should be keccak256 of EIP712 domain struct

    BitcoinTx
      validateProof
        when used with a valid but long proof
          ✓ should validate the proof with success
          ✓ should consume around 95000 gas

    Bridge - Deposit
  transferred 4500000000 T to the VendingMachine for KEEP
  transferred 4500000000 T to the VendingMachine for NU
  Warning: Potentially unsafe deployment of WalletRegistry

      You are using the `unsafeAllow.external-library-linking` flag to include external libraries.
      Make sure you have manually checked that the linked libraries are upgrade safe.

  Warning: Potentially unsafe deployment of BridgeStub

      You are using the `unsafeAllow.external-library-linking` flag to include external libraries.
      Make sure you have manually checked that the linked libraries are upgrade safe.

  Initialized Wallet Owner address: 0x3c705dB336C81c7FEFC5746e283aB2c0781A4B7b in transaction:
  0x4c54557085513b45258fe2a2f2b11d7b8abe6f870942f0d513209c4d26df7624
      revealDeposit
        when wallet is in Live state
          when reveal ahead period validation is disabled
            when funding transaction is P2SH
              when funding output script hash is correct
                when deposit was not revealed yet
                  when amount is not below the dust threshold
                    when deposit is routed to a trusted vault
                      ✓ should store proper deposit data
                      ✓ should emit DepositRevealed event
                    when deposit is not routed to a vault
                      ✓ should accept the deposit
                    when deposit treasury fee is zero
                      ✓ should store proper deposit data
                      ✓ should accept the deposit
                    when deposit is routed to a non-trusted vault
                      ✓ should revert
                  when amount is below the dust threshold
                    ✓ should revert
                when deposit was already revealed
                  ✓ should revert
              when funding output script hash is wrong
                ✓ should revert
              when the caller address does not match the funding output script
                ✓ should revert
              when funding transaction embeds extra data
                ✓ should revert
            when funding transaction is P2WSH
              when funding output script hash is correct
                when deposit was not revealed yet
                  when deposit is routed to a trusted vault
                    ✓ should store proper deposit data
                    ✓ should emit DepositRevealed event
                  when deposit is not routed to a vault
                    ✓ should accept the deposit
                  when deposit is routed to a non-trusted vault
                    ✓ should revert
                when deposit was already revealed
                  ✓ should revert
              when funding output script hash is wrong
                ✓ should revert
              when the caller address does not match the funding output script
                ✓ should revert
              when funding transaction embeds extra data
                ✓ should revert
            when funding transaction is neither P2SH nor P2WSH
              ✓ should revert
          when reveal ahead period validation is enabled
            when reveal ahead period is preserved
              ✓ should pass the refund locktime validation
            when reveal ahead period is not preserved
              ✓ should revert
            when refund locktime integer value is less than 500M
              ✓ should revert
        when wallet is not in Live state
          when wallet state is Unknown
            ✓ should revert
          when wallet state is MovingFunds
            ✓ should revert
          when the source wallet is in the Closing state
            ✓ should revert
          when wallet state is Closed
            ✓ should revert
          when wallet state is Terminated
            ✓ should revert
      revealDepositWithExtraData
        when extra data is non-zero
          when wallet is in Live state
            when reveal ahead period validation is disabled
              when funding transaction is P2SH
                when funding output script hash is correct
                  when deposit was not revealed yet
                    when amount is not below the dust threshold
                      when deposit is routed to a trusted vault
                        ✓ should store proper deposit data
                        ✓ should emit DepositRevealed event
                      when deposit is not routed to a vault
                        ✓ should accept the deposit
                      when deposit treasury fee is zero
                        ✓ should store proper deposit data
                        ✓ should accept the deposit
                      when deposit is routed to a non-trusted vault
                        ✓ should revert
                    when amount is below the dust threshold
                      ✓ should revert
                  when deposit was already revealed
                    ✓ should revert
                when funding output script hash is wrong
                  ✓ should revert
                when the caller address does not match the funding output script
                  ✓ should revert
                when the revealed extra data do not match
                  ✓ should revert
                when funding transaction does not embed extra data
                  ✓ should revert
              when funding transaction is P2WSH
                when funding output script hash is correct
                  when deposit was not revealed yet
                    when deposit is routed to a trusted vault
                      ✓ should store proper deposit data
                      ✓ should emit DepositRevealed event
                    when deposit is not routed to a vault
                      ✓ should accept the deposit
                    when deposit is routed to a non-trusted vault
                      ✓ should revert
                  when deposit was already revealed
                    ✓ should revert
                when funding output script hash is wrong
                  ✓ should revert
                when the caller address does not match the funding output script
                  ✓ should revert
                when the revealed extra data do not match
                  ✓ should revert
                when funding transaction does not embed extra data
                  ✓ should revert
              when funding transaction is neither P2SH nor P2WSH
                ✓ should revert
            when reveal ahead period validation is enabled
              when reveal ahead period is preserved
                ✓ should pass the refund locktime validation
              when reveal ahead period is not preserved
                ✓ should revert
              when refund locktime integer value is less than 500M
                ✓ should revert
          when wallet is not in Live state
            when wallet state is Unknown
              ✓ should revert
            when wallet state is MovingFunds
              ✓ should revert
            when the source wallet is in the Closing state
              ✓ should revert
            when wallet state is Closed
              ✓ should revert
            when wallet state is Terminated
              ✓ should revert
        when extra data is zero
          ✓ should revert
      submitDepositSweepProof
        when the wallet state is Live
          when transaction proof is valid
            when there is only one output
              when the single output is 20-byte
                when single output is either P2PKH or P2WPKH
                  when main UTXO data are valid
                    when transaction fee does not exceed the deposit transaction maximum fee
                      when there is only one input
                        when the single input is a revealed unswept P2SH deposit
                          ✓ should mark deposit as swept
                          ✓ should update main UTXO for the given wallet
                          ✓ should update the depositor's balance
                          ✓ should transfer collected treasury fee
                          ✓ should emit DepositsSwept event
                        when the single input is a revealed unswept P2WSH deposit
                          ✓ should mark deposit as swept
                          ✓ should update main UTXO for the given wallet
                          ✓ should update the depositor's balance
                          ✓ should transfer collected treasury fee
                          ✓ should emit DepositsSwept event
                        when the single input is a revealed unswept deposit with a trusted vault
                          ✓ should mark deposit as swept
                          ✓ should update main UTXO for the given wallet
                          ✓ should not update the depositor's balance
                          ✓ should update the vault's balance
                          ✓ should call the vault's receiveBalanceIncrease function
                          ✓ should transfer collected treasury fee
                          ✓ should emit DepositsSwept event
                        when the deposit treasury fee is zero
                          ✓ should update the depositor's balance
                          ✓ should collect no treasury fee
                        when the single input is a revealed unswept deposit with a non-trusted vault
                          ✓ should mark deposit as swept
                          ✓ should update main UTXO for the given wallet
                          ✓ should update the depositor's balance
                          ✓ should transfer collected treasury fee
                          ✓ should emit DepositsSwept event
                        when the single input is a revealed unswept deposit with a trusted vault but non-
  equal to the vault passed via function parameter
                          ✓ should revert
                        when the single input is the expected main UTXO
                          ✓ should revert
                        when the single input is a revealed but already swept deposit
                          ✓ should revert
                        when the single input is an unknown
                          ✓ should revert
                      when there are multiple inputs
                        when input vector consists only of revealed unswept deposits and the expected main
  UTXO
                          ✓ should mark deposits as swept
                          ✓ should update main UTXO for the given wallet
                          ✓ should update the depositors balances
                          ✓ should transfer collected treasury fee
                          ✓ should mark the previous main UTXO as spent
                          ✓ should emit DepositsSwept event
                        when input vector consists only of revealed unswept deposits with a trusted vault
  and the expected main UTXO
                          ✓ should mark deposits as swept
                          ✓ should update main UTXO for the given wallet
                          ✓ should not update the depositors balances
                          ✓ should update the vault's balance
                          ✓ should call the vault's receiveBalanceIncrease function
                          ✓ should transfer collected treasury fee
                          ✓ should mark the previous main UTXO as spent
                          ✓ should emit DepositsSwept event
                        when input vector consists only of revealed unswept deposits with a non-trusted
  vault and the expected main UTXO
                          ✓ should mark deposits as swept
                          ✓ should update main UTXO for the given wallet
```

```
                          ✓ should update the depositors balances
                          ✓ should transfer collected treasury fee
                          ✓ should mark the previous main UTXO as spent
                          ✓ should emit DepositsSwept event
                        when input vector consists only of revealed unswept deposits with different trusted
vaults and the expected main UTXO
                          ✓ should revert
                        when input vector consists only of revealed unswept deposits but there is no main
UTXO since it is not expected
                          ✓ should mark deposits as swept
                          ✓ should update main UTXO for the given wallet
                          ✓ should update the depositors balances
                          ✓ should transfer collected treasury fee
                          ✓ should emit DepositsSwept event
                        when input vector consists only of revealed unswept deposits but there is no main
UTXO despite it is expected
                          ✓ should revert
                        when input vector contains a revealed but already swept deposit
                          ✓ should revert
                        when input vector contains an unknown input
                          ✓ should revert
                    when transaction fee exceeds the deposit transaction maximum fee
                      ✓ should revert
                  when main UTXO data are invalid
                    ✓ should revert
                when single output is neither P2PKH nor P2WPKH
                  ✓ should revert
              when the single output is not 20-byte
                ✓ should revert
            when output count is other than one
              ✓ should revert
          when transaction proof is not valid
            when input vector is not valid
              ✓ should revert
            when output vector is not valid
              ✓ should revert
            when transaction is not on same level of merkle tree as coinbase
              ✓ should revert
            when merkle proof is not valid
              ✓ should revert
            when coinbase merkle proof is not valid
              ✓ should revert
            when proof difficulty is not current nor previous
              ✓ should revert
            when headers chain length is not valid
              ✓ should revert
            when headers chain is not valid
              ✓ should revert
            when the work in the header is insufficient
              ✓ should revert
            when accumulated difficulty in headers chain is insufficient
Warning: Potentially unsafe deployment of BridgeStub

    You are using the `unsafeAllow.external-library-linking` flag to include external libraries.
    Make sure you have manually checked that the linked libraries are upgrade safe.

              ✓ should revert
            when transaction data is limited to 64 bytes
              ✓ should revert
        when the wallet state is MovingFunds
          ✓ should succeed
        when the wallet state is neither Live or MovingFunds
          when wallet state is Unknown
            ✓ should revert
          when wallet state is Closing
            ✓ should revert
          when wallet state is Closed
            ✓ should revert
          when wallet state is Terminated
            ✓ should revert

  Bridge - Fraud
```

```
submitFraudChallenge
  when the wallet is in Live state
    when the amount of ETH deposited is enough
      when the data needed for signature verification is correct
        when the fraud challenge does not exist yet
          ✓ should transfer ether from the caller to the bridge
          ✓ should store the fraud challenge data
          ✓ should emit FraudChallengeSubmitted event
        when the fraud challenge already exists
          ✓ should revert
      when incorrect wallet public key is used
        ✓ should revert
      when incorrect sighash is used
        ✓ should revert
      when incorrect recovery ID is used
        ✓ should revert
      when incorrect signature data is used
        ✓ should revert
    when the amount of ETH deposited is too low
      ✓ should revert
  when the wallet is in MovingFunds state
    ✓ should succeed
  when the wallet is in Closing state
    ✓ should succeed
  when the wallet is in neither Live nor MovingFunds nor Closing state
    when wallet state is Unknown
      ✓ should revert
    when wallet state is Closed
      ✓ should revert
    when wallet state is Terminated
      ✓ should revert
defeatFraudChallengeWithHeartbeat
  when the challenge exists
    when the challenge is open
      when the heartbeat message has correct format
        ✓ should mark the challenge as resolved
        ✓ should send the ether deposited by the challenger to the treasury
        ✓ should emit FraudChallengeDefeated event
      when the heartbeat message has no correct format
        ✓ should revert
    when the challenge is resolved by defeat
      ✓ should revert
    when the challenge is resolved by timeout
      ✓ should revert
  when the challenge does not exist
    ✓ should revert
defeatFraudChallenge
  when the challenge exists
    when the challenge is open
      when the sighash type is correct
        when the input is non-witness
          when the transaction has single input
            when the input is marked as correctly spent in the Bridge
              ✓ should mark the challenge as resolved
              ✓ should send the ether deposited by the challenger to the treasury
              ✓ should emit FraudChallengeDefeated event
            when the input is not marked as correctly spent in the Bridge
              ✓ should revert
          when the transaction has multiple inputs
            when the input is marked as correctly spent in the Bridge
              ✓ should mark the challenge as resolved
              ✓ should send the ether deposited by the challenger to the treasury
              ✓ should emit FraudChallengeDefeated event
            when the input is not marked as correctly spent in the Bridge
              ✓ should revert
        when the input is witness
          when the transaction has single input
            when the input is marked as correctly spent in the Bridge
              ✓ should mark the challenge as resolved
              ✓ should send the ether deposited by the challenger to the treasury
              ✓ should emit FraudChallengeDefeated event
            when the input is not marked as correctly spent in the Bridge
```

```
              ✓ should revert
          when the transaction has multiple inputs
            when the input is marked as correctly spent in the Bridge
              ✓ should mark the challenge as resolved
              ✓ should send the ether deposited by the challenger to the treasury
              ✓ should emit FraudChallengeDefeated event
            when the input is not marked as correctly spent in the Bridge
              ✓ should revert
        when the sighash type is incorrect
          ✓ should revert
      when the challenge is resolved by defeat
        ✓ should revert
      when the challenge is resolved by timeout
        ✓ should revert
    when the challenge does not exist
      ✓ should revert
  notifyFraudChallengeDefeatTimeout
    when the fraud challenge exists
      when the fraud challenge is open
        when the fraud challenge has timed out
          when the wallet is in the Live or MovingFunds or Closing state
            when wallet state is Live but the wallet is not the active one
              ✓ should mark the fraud challenge as resolved
              ✓ should return the deposited ether to the challenger
              ✓ should emit FraudChallengeDefeatTimedOut event
              ✓ should change the wallet state to Terminated
              ✓ should emit WalletTerminated event
              ✓ should call the ECDSA wallet registry's closeWallet function
              ✓ should call the ECDSA wallet registry's seize function
              ✓ should decrease the live wallets count
              ✓ should not unset the active wallet
            when wallet state is Live and the wallet is the active one
              ✓ should mark the fraud challenge as resolved
              ✓ should return the deposited ether to the challenger
              ✓ should emit FraudChallengeDefeatTimedOut event
              ✓ should change the wallet state to Terminated
              ✓ should emit WalletTerminated event
              ✓ should call the ECDSA wallet registry's closeWallet function
              ✓ should call the ECDSA wallet registry's seize function
              ✓ should decrease the live wallets count
              ✓ should unset the active wallet
            when wallet state is MovingFunds
              ✓ should mark the fraud challenge as resolved
              ✓ should return the deposited ether to the challenger
              ✓ should emit FraudChallengeDefeatTimedOut event
              ✓ should change the wallet state to Terminated
              ✓ should emit WalletTerminated event
              ✓ should call the ECDSA wallet registry's closeWallet function
              ✓ should call the ECDSA wallet registry's seize function
            when wallet state is Closing
              ✓ should mark the fraud challenge as resolved
              ✓ should return the deposited ether to the challenger
              ✓ should emit FraudChallengeDefeatTimedOut event
              ✓ should change the wallet state to Terminated
              ✓ should emit WalletTerminated event
              ✓ should call the ECDSA wallet registry's closeWallet function
              ✓ should call the ECDSA wallet registry's seize function
          when the wallet is in the Terminated state
            ✓ should mark the fraud challenge as resolved
            ✓ should return the deposited ether to the challenger
            ✓ should emit FraudChallengeDefeatTimedOut event
            ✓ should not change the wallet state
            ✓ should not call the ECDSA wallet registry's seize function
          when the wallet is neither in the Live nor MovingFunds nor Closing nor Terminated state
            when the wallet is in the Unknown state
              ✓ should revert
            when the wallet is in the Closed state
              ✓ should revert
        when the fraud challenge has not timed out yet
          ✓ should revert
      when the fraud challenge is resolved by challenge defeat
        ✓ should revert
```

```
          when the fraud challenge is resolved by previous timeout notification
            ✓ should revert
        when the fraud challenge does not exist
          ✓ should revert

    Bridge - Governance
      beginGovernanceDelayUpdate
        when the caller is not the owner
          ✓ should revert
        when the caller is the owner
          ✓ should not update the governance delay
      finalizeGovernanceDelayUpdate
        when the caller is not the owner
          ✓ should revert
        when the update process is not initialized
          ✓ should revert
        when the governance delay has not passed
          ✓ should revert
        when the update process is initialized and governance delay passed
          ✓ should update the governance delay
          ✓ should reset the governance delay timer
      beginBridgeGovernanceTransfer
        when the caller is not the owner
          ✓ should revert
        when the caller is the owner
          ✓ should not update the bridge governance
          ✓ should not update the bridge governance owner
          ✓ should emit BridgeGovernanceTransferStarted event
      finalizeBridgeGovernanceTransfer
        when the caller is not the owner
          ✓ should revert
        when the update process is not initialized
          ✓ should revert
        when the governance delay has not passed
          ✓ should revert
        when the update process is initialized and governance delay passed
          ✓ should update the bridge governance
          ✓ should not update the bridgeGovernance owner
      beginDepositDustThresholdUpdate
        when the caller is not the owner
          ✓ should revert
        when the caller is the owner
          ✓ should not update the deposit dust threshold
          ✓ should emit DepositDustThresholdUpdateStarted event
      finalizeDepositDustThresholdUpdate
        when the caller is not the owner
          ✓ should revert
        when the update process is not initialized
          ✓ should revert
        when the governance delay has not passed
          ✓ should revert
        when the update process is initialized and governance delay passed
          ✓ should update the deposit dust threshold
          ✓ should emit DepositDustThresholdUpdated event
      beginDepositTreasuryFeeDivisorUpdate
        when the caller is not the owner
          ✓ should revert
        when the caller is the owner
          ✓ should not update the deposit treasury fee divisor
          ✓ should emit DepositTreasuryFeeDivisorUpdateStarted event
      finalizeDepositTreasuryFeeDivisorUpdate
        when the caller is not the owner
          ✓ should revert
        when the update process is not initialized
          ✓ should revert
        when the governance delay has not passed
          ✓ should revert
        when the update process is initialized and governance delay passed
          ✓ should update the deposit treasury fee divisor
          ✓ should emit DepositTreasuryFeeDivisorUpdated event
      beginDepositTxMaxFeeUpdate
        when the caller is not the owner
      ✓ should revert
    when the caller is the owner
      ✓ should not update the deposit tx max fee
      ✓ should emit DepositTxMaxFeeUpdateStarted event
  finalizeDepositTxMaxFeeUpdate
    when the caller is not the owner
      ✓ should revert
    when the update process is not initialized
      ✓ should revert
    when the governance delay has not passed
      ✓ should revert
    when the update process is initialized and governance delay passed
      ✓ should update the deposit tx max fee
      ✓ should emit DepositTxMaxFeeUpdated event
  beginDepositRevealAheadPeriodUpdate
    when the caller is not the owner
      ✓ should revert
    when the caller is the owner
      ✓ should not update the deposit reveal ahead period
      ✓ should emit DepositRevealAheadPeriodUpdateStarted event
  finalizeDepositRevealAheadPeriodUpdate
    when the caller is not the owner
      ✓ should revert
    when the update process is not initialized
      ✓ should revert
    when the governance delay has not passed
      ✓ should revert
    when the update process is initialized and governance delay passed
      ✓ should update the deposit reveal ahead period
      ✓ should emit DepositRevealAheadPeriodUpdated event
  beginRedemptionDustThresholdUpdate
    when the caller is not the owner
      ✓ should revert
    when the caller is the owner
      ✓ should not update the redemption dust threshold
      ✓ should emit RedemptionDustThresholdUpdateStarted event
  finalizeRedemptionDustThresholdUpdate
    when the caller is not the owner
      ✓ should revert
    when the update process is not initialized
      ✓ should revert
    when the governance delay has not passed
      ✓ should revert
    when the update process is initialized and governance delay passed
      ✓ should update the redemption dust threshold
      ✓ should emit RedemptionDustThresholdUpdated event
  beginRedemptionTreasuryFeeDivisorUpdate
    when the caller is not the owner
      ✓ should revert
    when the caller is the owner
      ✓ should not update the redemption treasury fee divisor
      ✓ should emit RedemptionTreasuryFeeDivisorUpdateStarted event
  finalizeRedemptionTreasuryFeeDivisorUpdate
    when the caller is not the owner
      ✓ should revert
    when the update process is not initialized
      ✓ should revert
    when the governance delay has not passed
      ✓ should revert
    when the update process is initialized and governance delay passed
      ✓ should update the redemption treasury fee divisor
      ✓ should emit RedemptionTreasuryFeeDivisorUpdated event
  beginRedemptionTxMaxTotalFeeUpdate
    when the caller is not the owner
      ✓ should revert
    when the caller is the owner
      ✓ should not update the redemption tx max total fee
      ✓ should emit RedemptionTxMaxTotalFeeUpdateStarted event
  finalizeRedemptionTxMaxTotalFeeUpdate
    when the caller is not the owner
      ✓ should revert
    when the update process is not initialized
```

```
      ✓ should revert
    when the governance delay has not passed
      ✓ should revert
    when the update process is initialized and governance delay passed
      ✓ should update the redemption tx max total fee
      ✓ should emit RedemptionTxMaxTotalFeeUpdated event
  beginRedemptionTimeoutUpdate
    when the caller is not the owner
      ✓ should revert
    when the caller is the owner
      ✓ should not update the redemption timeout
      ✓ should emit RedemptionTimeoutUpdateStarted event
  finalizeRedemptionTimeoutUpdate
    when the caller is not the owner
      ✓ should revert
    when the update process is not initialized
      ✓ should revert
    when the governance delay has not passed
      ✓ should revert
    when the update process is initialized and governance delay passed
      ✓ should update the redemption timeout
      ✓ should emit RedemptionTimeoutUpdated event
  beginRedemptionTimeoutSlashingAmountUpdate
    when the caller is not the owner
      ✓ should revert
    when the caller is the owner
      ✓ should not update the redemption timeout slashing amount
      ✓ should emit RedemptionTimeoutSlashingAmountUpdateStarted event
  finalizeRedemptionTimeoutSlashingAmountUpdate
    when the caller is not the owner
      ✓ should revert
    when the update process is not initialized
      ✓ should revert
    when the governance delay has not passed
      ✓ should revert
    when the update process is initialized and governance delay passed
      ✓ should update the redemption timeout slashing amount
      ✓ should emit RedemptionTimeoutSlashingAmountUpdated event
  beginRedemptionTimeoutNotifierRewardMultiplierUpdate
    when the caller is not the owner
      ✓ should revert
    when the caller is the owner
      ✓ should not update the redemption timeout notifier reward multiplier
      ✓ should emit RedemptionTimeoutNotifierRewardMultiplierUpdateStarted event
  finalizeRedemptionTimeoutNotifierRewardMultiplierUpdate
    when the caller is not the owner
      ✓ should revert
    when the update process is not initialized
      ✓ should revert
    when the governance delay has not passed
      ✓ should revert
    when the update process is initialized and governance delay passed
      ✓ should update the redemption timeout notifier reward multiplier
      ✓ should emit RedemptionTimeoutNotifierRewardMultiplierUpdated event
  beginMovingFundsTxMaxTotalFeeUpdate
    when the caller is not the owner
      ✓ should revert
    when the caller is the owner
      ✓ should not update the moving funds tx max total fee
      ✓ should emit MovingFundsTxMaxTotalFeeUpdateStarted event
  finalizeMovingFundsTxMaxTotalFeeUpdate
    when the caller is not the owner
      ✓ should revert
    when the update process is not initialized
      ✓ should revert
    when the governance delay has not passed
      ✓ should revert
    when the update process is initialized and governance delay passed
      ✓ should update the moving funds tx max total fee
      ✓ should emit MovingFundsTxMaxTotalFeeUpdated event
  beginMovingFundsDustThresholdUpdate
    when the caller is not the owner
```

```
      ✓ should revert
    when the caller is the owner
      ✓ should not update the moving funds dust threshold
      ✓ should emit MovingFundsDustThresholdUpdateStarted event
  finalizeMovingFundsDustThresholdUpdate
    when the caller is not the owner
      ✓ should revert
    when the update process is not initialized
      ✓ should revert
    when the governance delay has not passed
      ✓ should revert
    when the update process is initialized and governance delay passed
      ✓ should update the moving funds dust threshold
      ✓ should emit MovingFundsDustThresholdUpdated event
  beginMovingFundsTimeoutResetDelayUpdate
    when the caller is not the owner
      ✓ should revert
    when the caller is the owner
      ✓ should not update the moving funds timeout reset delay
      ✓ should emit MovingFundsTimeoutResetDelayUpdateStarted event
  finalizeMovingFundsTimeoutResetDelayUpdate
    when the caller is not the owner
      ✓ should revert
    when the update process is not initialized
      ✓ should revert
    when the governance delay has not passed
      ✓ should revert
    when the update process is initialized and governance delay passed
      ✓ should update the moving funds timeout reset delay
      ✓ should emit MovingFundsTimeoutResetDelayUpdated event
  beginMovingFundsTimeoutUpdate
    when the caller is not the owner
      ✓ should revert
    when the caller is the owner
      ✓ should not update the moving funds timeout
      ✓ should emit MovingFundsTimeoutUpdateStarted event
  finalizeMovingFundsTimeoutUpdate
    when the caller is not the owner
      ✓ should revert
    when the update process is not initialized
      ✓ should revert
    when the governance delay has not passed
      ✓ should revert
    when the update process is initialized and governance delay passed
      ✓ should update the moving funds timeout
      ✓ should emit MovingFundsTimeoutUpdated event
  beginMovingFundsTimeoutSlashingAmountUpdate
    when the caller is not the owner
      ✓ should revert
    when the caller is the owner
      ✓ should not update the moving funds timeout slashing amount
      ✓ should emit MovingFundsTimeoutSlashingAmountUpdateStarted event
  finalizeMovingFundsTimeoutSlashingAmountUpdate
    when the caller is not the owner
      ✓ should revert
    when the update process is not initialized
      ✓ should revert
    when the governance delay has not passed
      ✓ should revert
    when the update process is initialized and governance delay passed
      ✓ should update the moving funds timeout slashing amount
      ✓ should emit MovingFundsTimeoutSlashingAmountUpdated event
  beginMovingFundsTimeoutNotifierRewardMultiplierUpdate
    when the caller is not the owner
      ✓ should revert
    when the caller is the owner
      ✓ should not update the moving funds timeout notifier reward multiplier
      ✓ should emit MovingFundsTimeoutNotifierRewardMultiplierUpdateStarted event
  finalizeMovingFundsTimeoutNotifierRewardMultiplierUpdate
    when the caller is not the owner
      ✓ should revert
    when the update process is not initialized
```

```
      ✓ should revert
    when the governance delay has not passed
      ✓ should revert
    when the update process is initialized and governance delay passed
      ✓ should update the moving funds timeout notifier reward multiplier
      ✓ should emit MovingFundsTimeoutNotifierRewardMultiplierUpdated event
  beginMovingFundsCommitmentGasOffsetUpdate
    when the caller is not the owner
      ✓ should revert
    when the caller is the owner
      ✓ should not update the moving funds commitment gas offset
      ✓ should emit MovingFundsCommitmentGasOffsetUpdateStarted event
  finalizeMovingFundsCommitmentGasOffsetUpdate
    when the caller is not the owner
      ✓ should revert
    when the update process is not initialized
      ✓ should revert
    when the governance delay has not passed
      ✓ should revert
    when the update process is initialized and governance delay passed
      ✓ should update the moving funds commitment gas offset
      ✓ should emit MovingFundsCommitmentGasOffsetUpdated event
  beginMovedFundsSweepTxMaxTotalFeeUpdate
    when the caller is not the owner
      ✓ should revert
    when the caller is the owner
      ✓ should not update the moved funds sweep tx max total fee
      ✓ should emit MovedFundsSweepTxMaxTotalFeeUpdateStarted event
  finalizeMovedFundsSweepTxMaxTotalFeeUpdate
    when the caller is not the owner
      ✓ should revert
    when the update process is not initialized
      ✓ should revert
    when the governance delay has not passed
      ✓ should revert
    when the update process is initialized and governance delay passed
      ✓ should update the moved funds sweep tx max total fee
      ✓ should emit MovedFundsSweepTxMaxTotalFeeUpdated event
  beginMovedFundsSweepTimeoutUpdate
    when the caller is not the owner
      ✓ should revert
    when the caller is the owner
      ✓ should not update the moved funds sweep timeout
      ✓ should emit MovedFundsSweepTimeoutUpdateStarted event
  finalizeMovedFundsSweepTimeoutUpdate
    when the caller is not the owner
      ✓ should revert
    when the update process is not initialized
      ✓ should revert
    when the governance delay has not passed
      ✓ should revert
    when the update process is initialized and governance delay passed
      ✓ should update the moved funds sweep timeout
      ✓ should emit MovedFundsSweepTimeoutUpdated event
  beginMovedFundsSweepTimeoutSlashingAmountUpdate
    when the caller is not the owner
      ✓ should revert
    when the caller is the owner
      ✓ should not update the moved funds sweep timeout slashing amount
      ✓ should emit MovedFundsSweepTimeoutSlashingAmountUpdateStarted event
  finalizeMovedFundsSweepTimeoutSlashingAmountUpdate
    when the caller is not the owner
      ✓ should revert
    when the update process is not initialized
      ✓ should revert
    when the governance delay has not passed
      ✓ should revert
    when the update process is initialized and governance delay passed
      ✓ should update the moved funds sweep timeout slashing amount
      ✓ should emit MovedFundsSweepTimeoutSlashingAmountUpdated event
  beginMovedFundsSweepTimeoutNotifierRewardMultiplierUpdate
    when the caller is not the owner
```

```
      ✓ should revert
    when the caller is the owner
      ✓ should not update the moved funds sweep timeout notifier reward multiplier
      ✓ should emit MovedFundsSweepTimeoutNotifierRewardMultiplierUpdateStarted event
  finalizeMovedFundsSweepTimeoutNotifierRewardMultiplierUpdate
    when the caller is not the owner
      ✓ should revert
    when the update process is not initialized
      ✓ should revert
    when the governance delay has not passed
      ✓ should revert
    when the update process is initialized and governance delay passed
      ✓ should update the moved funds sweep timeout notifier reward multiplier
      ✓ should emit MovedFundsSweepTimeoutNotifierRewardMultiplierUpdated event
  beginWalletCreationPeriodUpdate
    when the caller is not the owner
      ✓ should revert
    when the caller is the owner
      ✓ should not update the wallet creation period
      ✓ should emit WalletCreationPeriodUpdateStarted event
  finalizeWalletCreationPeriodUpdate
    when the caller is not the owner
      ✓ should revert
    when the update process is not initialized
      ✓ should revert
    when the governance delay has not passed
      ✓ should revert
    when the update process is initialized and governance delay passed
      ✓ should update the wallet creation period
      ✓ should emit WalletCreationPeriodUpdated event
  beginWalletCreationMinBtcBalanceUpdate
    when the caller is not the owner
      ✓ should revert
    when the caller is the owner
      ✓ should not update the wallet creation min btc balance
      ✓ should emit WalletCreationMinBtcBalanceUpdateStarted event
  finalizeWalletCreationMinBtcBalanceUpdate
    when the caller is not the owner
      ✓ should revert
    when the update process is not initialized
      ✓ should revert
    when the governance delay has not passed
      ✓ should revert
    when the update process is initialized and governance delay passed
      ✓ should update the wallet creation min btc balance
      ✓ should emit WalletCreationMinBtcBalanceUpdated event
  beginWalletCreationMaxBtcBalanceUpdate
    when the caller is not the owner
      ✓ should revert
    when the caller is the owner
      ✓ should not update the wallet creation max btc balance
      ✓ should emit WalletCreationMaxBtcBalanceUpdateStarted event
  finalizeWalletCreationMaxBtcBalanceUpdate
    when the caller is not the owner
      ✓ should revert
    when the update process is not initialized
      ✓ should revert
    when the governance delay has not passed
      ✓ should revert
    when the update process is initialized and governance delay passed
      ✓ should update the wallet creation max btc balance
      ✓ should emit WalletCreationMaxBtcBalanceUpdated event
  beginWalletClosureMinBtcBalanceUpdate
    when the caller is not the owner
      ✓ should revert
    when the caller is the owner
      ✓ should not update the wallet closure min btc balance
      ✓ should emit WalletClosureMinBtcBalanceUpdateStarted event
  finalizeWalletClosureMinBtcBalanceUpdate
    when the caller is not the owner
      ✓ should revert
    when the update process is not initialized
```

```
        ✓ should revert
      when the governance delay has not passed
        ✓ should revert
      when the update process is initialized and governance delay passed
        ✓ should update the wallet closure min btc balance
        ✓ should emit WalletClosureMinBtcBalanceUpdated event
    beginWalletMaxAgeUpdate
      when the caller is not the owner
        ✓ should revert
      when the caller is the owner
        ✓ should not update the wallet max age
        ✓ should emit WalletMaxAgeUpdateStarted event
    finalizeWalletMaxAgeUpdate
      when the caller is not the owner
        ✓ should revert
      when the update process is not initialized
        ✓ should revert
      when the governance delay has not passed
        ✓ should revert
      when the update process is initialized and governance delay passed
        ✓ should update the wallet max age
        ✓ should emit WalletMaxAgeUpdated event
    beginWalletMaxBtcTransferUpdate
      when the caller is not the owner
        ✓ should revert
      when the caller is the owner
        ✓ should not update the wallet max btc transfer
        ✓ should emit WalletMaxBtcTransferUpdateStarted event
    finalizeWalletMaxBtcTransferUpdate
      when the caller is not the owner
        ✓ should revert
      when the update process is not initialized
        ✓ should revert
      when the governance delay has not passed
        ✓ should revert
      when the update process is initialized and governance delay passed
        ✓ should update the wallet max btc transfer
        ✓ should emit WalletMaxBtcTransferUpdated event
    beginWalletClosingPeriodUpdate
      when the caller is not the owner
        ✓ should revert
      when the caller is the owner
        ✓ should not update the wallet closing period
        ✓ should emit WalletClosingPeriodUpdateStarted event
    finalizeWalletClosingPeriodUpdate
      when the caller is not the owner
        ✓ should revert
      when the update process is not initialized
        ✓ should revert
      when the governance delay has not passed
        ✓ should revert
      when the update process is initialized and governance delay passed
        ✓ should update the wallet closing period
        ✓ should emit WalletClosingPeriodUpdated event
    beginFraudChallengeDepositAmountUpdate
      when the caller is not the owner
        ✓ should revert
      when the caller is the owner
        ✓ should not update the fraud challenge deposit amount
        ✓ should emit FraudChallengeDepositAmountUpdateStarted event
    finalizeFraudChallengeDepositAmountUpdate
      when the caller is not the owner
        ✓ should revert
      when the update process is not initialized
        ✓ should revert
      when the governance delay has not passed
        ✓ should revert
      when the update process is initialized and governance delay passed
        ✓ should update the fraud challenge deposit amount
        ✓ should emit FraudChallengeDepositAmountUpdated event
    beginFraudChallengeDefeatTimeoutUpdate
      when the caller is not the owner
      ✓ should revert
    when the caller is the owner
      ✓ should not update the fraud challenge defeat timeout
      ✓ should emit FraudChallengeDefeatTimeoutUpdateStarted event
  finalizeFraudChallengeDefeatTimeoutUpdate
    when the caller is not the owner
      ✓ should revert
    when the update process is not initialized
      ✓ should revert
    when the governance delay has not passed
      ✓ should revert
    when the update process is initialized and governance delay passed
      ✓ should update the fraud challenge defeat timeout
      ✓ should emit FraudChallengeDefeatTimeoutUpdated event
  beginFraudSlashingAmountUpdate
    when the caller is not the owner
      ✓ should revert
    when the caller is the owner
      ✓ should not update the fraud slashing amount
      ✓ should emit FraudSlashingAmountUpdateStarted event
  finalizeFraudSlashingAmountUpdate
    when the caller is not the owner
      ✓ should revert
    when the update process is not initialized
      ✓ should revert
    when the governance delay has not passed
      ✓ should revert
    when the update process is initialized and governance delay passed
      ✓ should update the fraud slashing amount
      ✓ should emit FraudSlashingAmountUpdated event
  beginFraudNotifierRewardMultiplierUpdate
    when the caller is not the owner
      ✓ should revert
    when the caller is the owner
      ✓ should not update the fraud notifier reward multiplier
      ✓ should emit FraudNotifierRewardMultiplierUpdateStarted event
  finalizeFraudNotifierRewardMultiplierUpdate
    when the caller is not the owner
      ✓ should revert
    when the update process is not initialized
      ✓ should revert
    when the governance delay has not passed
      ✓ should revert
    when the update process is initialized and governance delay passed
      ✓ should update the fraud notifier reward multiplier
      ✓ should emit FraudNotifierRewardMultiplierUpdated event
  beginTreasuryUpdate
    when the caller is not the owner
      ✓ should revert
    when the caller is the owner
      ✓ should not update the treasury address
      ✓ should emit TreasuryUpdateStarted event
  finalizeTreasuryUpdate
    when the caller is not the owner
      ✓ should revert
    when the update process is not initialized
      ✓ should revert
    when the governance delay has not passed
      ✓ should revert
    when the update process is initialized and governance delay passed
      ✓ should update the treasury address
      ✓ should emit TreasuryUpdated event
  setVaultStatus
    when the caller is not the owner
      ✓ should revert
    when the caller is the owner
      ✓ should mark the vault as trusted
      ✓ should emit VaultStatusUpdated event
  setRedemptionWatchtower
    when caller is not the owner
      ✓ should revert
    when caller is the owner
```

```
        ✓ should not revert

  Bridge - Moving funds
    submitMovingFundsCommitment
      when source wallet is in the MovingFunds state
        when source wallet has no pending redemptions
          when source wallet has no pending moved funds sweep requests
            when the commitment was not submitted yet
              when the caller is a member of the source wallet
                when passed wallet main UTXO is valid
                  when wallet balance is greater than zero
                    when the expected target wallets count is greater than zero
                      when the submitted target wallets count is same as the expected
                        when all target wallets are different than the source wallet
                          when all target wallets follow the expected order
                            when all target wallets are in the Live state
                              ✓ should store the target wallets commitment for the given wallet
                              ✓ should emit the MovingFundsCommitmentSubmitted event
                              ✓ should refund ETH
                            when one of the target wallets is not in the Live state
                              ✓ should revert
                          when one of the target wallets break the expected order
                            ✓ should revert
                        when one of the target wallets is same as the source wallet
                          ✓ should revert
                      when the submitted target wallets count is other than the expected
                        ✓ should revert
                    when the expected target wallets count is zero
                      ✓ should revert
                  when wallet balance is zero
                    ✓ should revert
                when passed wallet main UTXO is invalid
                  ✓ should revert
              when the caller is not a member of the source wallet
                ✓ should revert
            when the commitment was already submitted
              ✓ should revert
          when source wallet has pending moved funds sweep requests
            ✓ should revert
        when source wallet has pending redemptions
          ✓ should revert
      when source wallet is not in the MovingFunds state
        when the source wallet is in the Unknown state
          ✓ should revert
        when the source wallet is in the Live state
          ✓ should revert
        when the source wallet is in the Closing state
          ✓ should revert
        when the source wallet is in the Closed state
          ✓ should revert
        when the source wallet is in the Terminated state
          ✓ should revert
    resetMovingFundsTimeout
      when the wallet is in the MovingFunds state
        when the wallet's commitment is not submitted yet
          when Live wallets count is zero
            when reset delay has elapsed
              ✓ should reset the moving funds timeout
              ✓ should emit MovingFundsTimeoutReset event
            when reset delay has not elapsed yet
              ✓ should revert
            when one reset occurred and the reset delay has elapsed again
              ✓ should reset the moving funds timeout
              ✓ should emit MovingFundsTimeoutReset event
            when one reset occurred and the reset delay has not elapsed yet
              ✓ should revert
          when Live wallets count is not zero
            ✓ should revert
        when the wallet's commitment is already submitted
          ✓ should revert
      when the wallet is not in the MovingFunds state
        when the wallet is in the Unknown state
```

```
          ✓ should revert
        when the wallet is in the Live state
          ✓ should revert
        when the wallet is in the Closing state
          ✓ should revert
        when the wallet is in the Closed state
          ✓ should revert
        when the wallet is in the Terminated state
          ✓ should revert
    submitMovingFundsProof
      when transaction proof is valid
        when there is a main UTXO for the given wallet
          when main UTXO data are valid
            when there is only one input
              when the single input points to the wallet's main UTXO
                when the output vector references only 20-byte hashes
                  when the output vector has only P2PKH and P2WPKH outputs
                    when transaction amount is distributed evenly
                      when transaction fee is not too high
                        when source wallet is in the MovingFunds state
                          when target wallets commitment is submitted
                            when actual target wallets correspond to the commitment
                              when there is a single target wallet
                                ✓ should mark the main UTXO as correctly spent
                                ✓ should unset the main UTXO for the source wallet
                                ✓ should put the source wallet in the Closing state
                                ✓ should set the closing started timestamp
                                ✓ should emit the WalletClosing event
                                ✓ should emit the MovingFundsCompleted event
                                ✓ should create appropriate moved funds sweep requests
                              when there are multiple target wallets and the amount is indivisible
                                ✓ should mark the main UTXO as correctly spent
                                ✓ should unset the main UTXO for the source wallet
                                ✓ should put the source wallet in the Closing state
                                ✓ should set the closing started timestamp
                                ✓ should emit the WalletClosing event
                                ✓ should emit the MovingFundsCompleted event
                                ✓ should create appropriate moved funds sweep requests
                              when there are multiple target wallets and the amount is divisible
                                ✓ should mark the main UTXO as correctly spent
                                ✓ should unset the main UTXO for the source wallet
                                ✓ should put the source wallet in the Closing state
                                ✓ should set the closing started timestamp
                                ✓ should emit the WalletClosing event
                                ✓ should emit the MovingFundsCompleted event
                                ✓ should create appropriate moved funds sweep requests
                            when actual target wallets does not correspond to the commitment
                              when funds were sent to more wallets than submitted in the commitment
                                ✓ should revert
                              when funds were sent to less wallets than submitted in the commitment
                                ✓ should revert
(node:4571) PromiseRejectionHandledWarning: Promise rejection was handled asynchronously (rejection id:
11)
(Use `node --trace-warnings ...` to show where the warning was created)
                              when funds were sent to completely different wallets than submitted in the
commitment
                                ✓ should revert
                              when funds were sent to the wallets submitted in the commitment but with a
wrong order
                                ✓ should revert
                          when target wallets commitment is not submitted
                            ✓ should revert
                        when source wallet is not in the MovingFunds state
                          when wallet state is Unknown
                            ✓ should revert
                          when wallet state is Live
                            ✓ should revert
                          when wallet state is Closing
                            ✓ should revert
                          when wallet state is Closed
                            ✓ should revert
                          when wallet state is Terminated
                            ✓ should revert
                      when transaction fee is too high
                        ✓ should revert
                    when transaction amount is not distributed evenly
                      ✓ should revert
                  when the output vector contains P2SH output
                    ✓ should revert
                when the output vector does not only reference 20-byte hashes
                  ✓ should revert
              when the single input doesn't point to the wallet's main UTXO
                ✓ should revert
            when input count is other than one
              ✓ should revert
          when main UTXO data are invalid
            ✓ should revert
        when there is no main UTXO for the given wallet
          ✓ should revert
      when transaction proof is not valid
        when input vector is not valid
          ✓ should revert
        when output vector is not valid
          ✓ should revert
        when transaction is not on same level of merkle tree as coinbase
          ✓ should revert
        when merkle proof is not valid
          ✓ should revert
        when coinbase merkle proof is not valid
          ✓ should revert
        when proof difficulty is not current nor previous
          ✓ should revert
        when headers chain length is not valid
          ✓ should revert
        when headers chain is not valid
          ✓ should revert
        when the work in the header is insufficient
          ✓ should revert
        when accumulated difficulty in headers chain is insufficient
Warning: Potentially unsafe deployment of BridgeStub

    You are using the `unsafeAllow.external-library-linking` flag to include external libraries.
    Make sure you have manually checked that the linked libraries are upgrade safe.

          ✓ should revert
        when transaction data is limited to 64 bytes
          ✓ should revert
    notifyMovingFundsTimeout
      when source wallet is in the MovingFunds state
        when the moving funds process has timed out
          ✓ should switch the wallet to Terminated state
          ✓ should emit WalletTerminated event
          ✓ should call ECDSA Wallet Registry's closeWallet function
          ✓ should call the ECDSA wallet registry's seize function
          ✓ should emit MovingFundsTimedOut event
        when the moving funds process has not timed out
          ✓ should revert
      when source wallet is not in the MovingFunds state
        when the source wallet is in the Unknown state
          ✓ should revert
        when the source wallet is in the Live state
          ✓ should revert
        when the source wallet is in the Closing state
          ✓ should revert
        when the source wallet is in the Closed state
          ✓ should revert
        when the source wallet is in the Terminated state
          ✓ should revert
    notifyMovingFundsBelowDust
      when the wallet is in the MovingFunds state
        when the main UTXO parameter is valid
          when the balance is below the dust threshold
            ✓ should change wallet's state to Closing
            ✓ should set the wallet's closing started timestamp
```

```
            ✓ should emit WalletClosing event
            ✓ should emit MovingFundsBelowDustReported event
          when the balance is not below the dust threshold
            ✓ should revert
        when the main UTXO parameter is invalid
          ✓ should revert
      when the wallet is not in the MovingFunds state
        when wallet state is Unknown
          ✓ should revert
        when wallet state is Live
          ✓ should revert
        when wallet state is Closing
          ✓ should revert
        when wallet state is Closed
          ✓ should revert
        when wallet state is Terminated
          ✓ should revert
    submitMovedFundsSweepProof
      when transaction proof is valid
        when there is only one output
          when the single output is 20-byte
            when single output is either P2PKH or P2WPKH
              when sweeping wallet is either in the Live or MovingFunds state
                when sweeping wallet is in the Live state
                  when main UTXO data are valid
                    when transaction fee does not exceed the sweep transaction maximum fee
                      when the sweeping wallet has no main UTXO set
                        when there is a single input referring to a Pending sweep request
                          ✓ should mark the sweep request as processed
                          ✓ should decrease the sweeping wallet's pending requests count
                          ✓ should set the transaction output as new sweeping wallet main UTXO
                          ✓ should emit the MovedFundsSwept event
                        when the single input does not refer to a Pending sweep request
                          when the single input refers to an Unknown sweep request
                            ✓ should revert
                          when the single input refers to a Processed sweep request
                            ✓ should revert
                          when the single input refers to a TimedOut sweep request
                            ✓ should revert
                        when the single input does refer to a Pending sweep request that belongs to
another wallet
                          ✓ should revert
                        when the number of inputs is other than one
                          ✓ should revert
                      when the sweeping wallet has a main UTXO set
                        when the first input refers to a Pending sweep request and the second input
refers to the sweeping wallet main UTXO
                          ✓ should mark the sweep request as processed
                          ✓ should decrease the sweeping wallet's pending requests count
                          ✓ should set the transaction output as new sweeping wallet main UTXO
                          ✓ should emit the MovedFundsSwept event
                          ✓ should mark the current sweeping wallet main UTXO as correctly spent
                        when the first input refers to the sweeping wallet main UTXO and the second input
refers to a Pending sweep request
                          ✓ should revert
                        when the first input does not refer to a Pending sweep request and the second
input refers to the sweeping wallet main UTXO
                          when the first input refers to an Unknown sweep request
                            ✓ should revert
                          when the first input refers to a Processed sweep request
                            ✓ should revert
(node:4571) PromiseRejectionHandledWarning: Promise rejection was handled asynchronously (rejection id:
12)
                          when the first input refers to a TimedOut sweep request
                            ✓ should revert
                        when the first input refers to a Pending sweep request that belongs to another
wallet and the second input refers to the sweeping wallet main UTXO
                          ✓ should revert
                        when the first input refers to a Pending sweep request and the second input does
not refer to the sweeping wallet main UTXO
                          ✓ should revert
                        when the number of inputs is other than two
```

```
                              ✓ should revert
                        when transaction fee exceeds the sweep transaction maximum fee
                          ✓ should revert
                      when main UTXO data are invalid
                        ✓ should revert
                    when sweeping wallet is in the MovingFunds state
                      ✓ should succeed
                  when sweeping wallet is neither in the Live nor MovingFunds state
                    when sweeping wallet is in the Unknown state
                      ✓ should revert
                    when sweeping wallet is in the Closing state
                      ✓ should revert
                    when sweeping wallet is in the Closed state
                      ✓ should revert
                    when sweeping wallet is in the Terminated state
                      ✓ should revert
                when single output is neither P2PKH nor P2WPKH
                  ✓ should revert
              when the single output is not 20-byte
                ✓ should revert
            when output count is other than one
              ✓ should revert
          when transaction proof is not valid
            when input vector is not valid
              ✓ should revert
            when output vector is not valid
              ✓ should revert
            when transaction is not on same level of merkle tree as coinbase
              ✓ should revert
            when merkle proof is not valid
              ✓ should revert
            when coinbase merkle proof is not valid
              ✓ should revert
            when proof difficulty is not current nor previous
              ✓ should revert
            when headers chain length is not valid
              ✓ should revert
            when headers chain is not valid
              ✓ should revert
            when the work in the header is insufficient
              ✓ should revert
            when accumulated difficulty in headers chain is insufficient
  Warning: Potentially unsafe deployment of BridgeStub

      You are using the `unsafeAllow.external-library-linking` flag to include external libraries.
      Make sure you have manually checked that the linked libraries are upgrade safe.

              ✓ should revert
            when transaction data is limited to 64 bytes
              ✓ should revert
        notifyMovedFundsSweepTimeout
          when moved funds sweep request is in the Pending state
            when moved funds sweep request has timed out
              when the wallet is either in the Live or MovingFunds state
                when the wallet is in the Live state but the wallet is not the active one
                  ✓ should switch the moved funds sweep request to the TimedOut state
                  ✓ should decrease the number of pending moved funds sweep requests for the given wallet
                  ✓ should switch the wallet to Terminated state
                  ✓ should emit WalletTerminated event
                  ✓ should call ECDSA Wallet Registry's closeWallet function
                  ✓ should call the ECDSA wallet registry's seize function
                  ✓ should emit MovedFundsSweepTimedOut event
                  ✓ should decrease the live wallets count
                  ✓ should not unset the active wallet
                when the wallet is in the Live state and the wallet is the active one
                  ✓ should switch the moved funds sweep request to the TimedOut state
                  ✓ should decrease the number of pending moved funds sweep requests for the given wallet
                  ✓ should switch the wallet to Terminated state
                  ✓ should emit WalletTerminated event
                  ✓ should call ECDSA Wallet Registry's closeWallet function
                  ✓ should call the ECDSA wallet registry's seize function
                  ✓ should emit MovedFundsSweepTimedOut event
                  ✓ should decrease the live wallets count
                  ✓ should unset the active wallet
                when the wallet is in the MovingFunds state
                  ✓ should switch the moved funds sweep request to the TimedOut state
                  ✓ should decrease the number of pending moved funds sweep requests for the given wallet
                  ✓ should switch the wallet to Terminated state
                  ✓ should emit WalletTerminated event
                  ✓ should call ECDSA Wallet Registry's closeWallet function
                  ✓ should call the ECDSA wallet registry's seize function
                  ✓ should emit MovedFundsSweepTimedOut event
              when the wallet is in the Terminated state
                ✓ should switch the moved funds sweep request to the TimedOut state
                ✓ should decrease the number of pending moved funds sweep requests for the given wallet
                ✓ should not change the wallet state
              when the wallet is neither in the Live nor MovingFunds nor Terminated state
                when the wallet is in the Unknown state
                  ✓ should revert
                when the wallet is in the Closing state
                  ✓ should revert
                when the wallet is in the Closed state
                  ✓ should revert
            when moved funds sweep request has not timed out yet
              ✓ should revert
          when moved funds sweep request is not in the Pending state
            when moved funds sweep request is in the Unknown state
              ✓ should revert
            when moved funds sweep request is in the Processed state
              ✓ should revert
            when moved funds sweep request is in the TimedOut state
              ✓ should revert

      Bridge - Parameters
        updateDepositParameters
          when caller is the contract guvnor
            when all new parameter values are correct
              ✓ should set correct values
              ✓ should emit DepositParametersUpdated event
              ✓ should emit DepositParametersUpdated event
              ✓ should emit DepositParametersUpdated event
              ✓ should emit DepositParametersUpdated event
            when new deposit dust threshold is zero
              ✓ should revert
            when new deposit dust threshold is same as deposit TX max fee
              ✓ should revert
            when new deposit dust threshold is lower than deposit TX max fee
              ✓ should revert
            when new deposit transaction max fee is zero
              ✓ should revert
          when caller is not the contract guvnor
            ✓ should revert
            ✓ should revert
            ✓ should revert
            ✓ should revert
        updateRedemptionParameters
          when caller is the contract guvnor
            when all new parameter values are correct
              ✓ should set correct values
              ✓ should emit RedemptionParametersUpdated event
              ✓ should emit RedemptionParametersUpdated event
              ✓ should emit RedemptionParametersUpdated event
              ✓ should emit RedemptionParametersUpdated event
              ✓ should emit RedemptionParametersUpdated event
              ✓ should emit RedemptionParametersUpdated event
              ✓ should emit RedemptionParametersUpdated event
            when new redemption dust threshold is not greater than moving funds dust threshold
              ✓ should revert
            when new redemption dust threshold is same as redemption tx max fee
              ✓ should revert
            when new redemption dust threshold is lower than redemption tx max fee
              ✓ should revert
            when new redemption transaction max fee is zero
              ✓ should revert
              when new redemption transaction max total fee is lesser than the redemption transaction per-
      request max fee
                ✓ should revert
              when new redemption timeout is zero
                ✓ should revert
              when new redemption timeout notifier reward multiplier is greater than 100
                ✓ should revert
            when caller is not the contract guvnor
              ✓ should revert
              ✓ should revert
              ✓ should revert
              ✓ should revert
              ✓ should revert
              ✓ should revert
              ✓ should revert
          updateMovingFundsParameters
            when caller is the contract guvnor
              when all new parameter values are correct
                ✓ should set correct values
                ✓ should emit MovingFundsParametersUpdated event
                ✓ should emit MovingFundsParametersUpdated event
                ✓ should emit MovingFundsParametersUpdated event
                ✓ should emit MovingFundsParametersUpdated event
                ✓ should emit MovingFundsParametersUpdated event
                ✓ should emit MovingFundsParametersUpdated event
                ✓ should emit MovingFundsParametersUpdated event
                ✓ should emit MovingFundsParametersUpdated event
                ✓ should emit MovingFundsParametersUpdated event
                ✓ should emit MovingFundsParametersUpdated event
                ✓ should emit MovingFundsParametersUpdated event
              when new moving funds transaction max total fee is zero
                ✓ should revert
              when new moving funds dust threshold is zero
                ✓ should revert
              when new moving funds dust threshold is not lower than redemption dust threshold
                ✓ should revert
              when new moving funds timeout reset delay is zero
                ✓ should revert
              when new moving funds timeout is not greater than its reset delay
                ✓ should revert
              when new moved funds sweep timeout is zero
                ✓ should revert
              when new moved funds sweep timeout notifier reward multiplier is greater than 100
                ✓ should revert
            when caller is not the contract guvnor
              ✓ should revert
          updateWalletParameters
            when caller is the contract guvnor
              when all new parameter values are correct
                ✓ should set correct values
                ✓ should emit WalletParametersUpdated event
                ✓ should emit WalletParametersUpdated event
                ✓ should emit WalletParametersUpdated event
                ✓ should emit WalletParametersUpdated event
                ✓ should emit WalletParametersUpdated event
                ✓ should emit WalletParametersUpdated event
                ✓ should emit WalletParametersUpdated event
              when new creation maximum BTC balance is not greater than the creation minimum BTC balance
                ✓ should revert
              when new maximum BTC transfer is zero
                ✓ should revert
              when new closing period is zero
                ✓ should revert
            when caller is not the contract guvnor
              ✓ should revert
          updateFraudParameters
            when caller is the contract guvnor
              when all new parameter values are correct
                ✓ should set correct values
                ✓ should emit FraudParametersUpdated event
                ✓ should emit FraudParametersUpdated event
                ✓ should emit FraudParametersUpdated event
                ✓ should emit FraudParametersUpdated event
              when new fraud challenge defeat timeout is zero
                ✓ should revert
              when new fraud notifier reward multiplier is greater than 100
                ✓ should revert
            when caller is not the contract guvnor
              ✓ should revert
          updateTreasury
            when caller is the contract guvnor
              when the new treasury address is non-zero
                ✓ should set the new treasury address
                ✓ should emit TreasuryUpdated event
              when the new treasury address is zero
                ✓ should revert
            when caller is not the contract guvnor
              ✓ should revert
          setRedemptionWatchtower
            when caller is not the contract guvnor
              ✓ should revert
            when caller is the contract guvnor
              when the watchtower address is already set
                ✓ should revert
              when the watchtower address is not set yet
                when the watchtower address is zero
                  ✓ should revert
                when the watchtower address is non-zero
                  ✓ should set the watchtower address
                  ✓ should emit RedemptionWatchtowerSet event

        Bridge - Redemption
          requestRedemption
            when redemption watchtower is not set
              when wallet state is Live
                when there is a main UTXO for the given wallet
                  when main UTXO data are valid
                    when redeemer output script is standard type
                      when redeemer output script does not point to the wallet public key hash
                        when amount is not below the dust threshold
                          when there is no pending request for the given redemption key
                            when wallet has sufficient funds
                              when redeemer made a sufficient allowance in Bank
                                when redeemer output script is P2WPKH
                                  ✓ should increase the wallet's pending redemptions value
                                  ✓ should store the redemption request
                                  ✓ should emit RedemptionRequested event
                                  ✓ should take the right balance from Bank
                                when redeemer output script is P2WSH
                                  ✓ should succeed
                                when redeemer output script is P2PKH
                                  ✓ should succeed
                                when redeemer output script is P2SH
                                  ✓ should succeed
                                when redemption treasury fee is zero
                                  ✓ should store the redemption request with zero fee
                              when redeemer has not made a sufficient allowance in Bank
                                ✓ should revert
                            when wallet has insufficient funds
                              ✓ should revert
                          when there is a pending request for the given redemption key
                            ✓ should revert
                        when amount is below the dust threshold
                          ✓ should revert
                      when redeemer output script points to the wallet public key hash
                        ✓ should revert
                    when redeemer output script is not standard type
                      ✓ should revert
                  when main UTXO data are invalid
                    ✓ should revert
                when there is no main UTXO for the given wallet
                  ✓ should revert
              when wallet state is other than Live
                when wallet state is Unknown
```

```
            ✓ should revert
          when wallet state is MovingFunds
            ✓ should revert
          when wallet state is Closing
            ✓ should revert
          when wallet state is Closed
            ✓ should revert
          when wallet state is Terminated
            ✓ should revert
      when redemption watchtower is set
        when redemption watchtower considers the redemption as unsafe
          ✓ should revert
        when redemption watchtower considers the redemption as safe
          ✓ should not revert
    receiveBalanceApproval
      when called via Bank.approveBalanceAndCall
        when wallet state is Live
          when there is a main UTXO for the given wallet
            when main UTXO data are valid
              when redeemer output script is standard type
                when redeemer output script does not point to the wallet public key hash
                  when amount is not below the dust threshold
                    when redeemer output script is P2WPKH
                      ✓ should increase the wallet's pending redemptions value
                      ✓ should store the redemption request
                      ✓ should emit RedemptionRequested event
                      ✓ should take the right balance from Bank
      when called directly
        ✓ should revert
    submitRedemptionProof
      when transaction proof is valid
        when there is a main UTXO for the given wallet
          when main UTXO data are valid
            when there is only one input
              when the single input points to the wallet's main UTXO
                when wallet state is Live
                  when the total transaction fee is not too high
                    when there is only one output
                      when the single output is a pending requested redemption
                        ✓ should close processed redemption request
                        ✓ should delete the wallet's main UTXO
                        ✓ should mark the previous main UTXO as spent
                        ✓ should decrease the wallet's pending redemptions value
                        ✓ should decrease Bridge's balance in Bank
                        ✓ should not transfer anything to the treasury
                        ✓ should not change redeemer balance in any way
                      when the single output is a non-reported timed out requested redemption
                        ✓ should close processed redemption request
                        ✓ should delete the wallet's main UTXO
                        ✓ should mark the previous main UTXO as spent
                        ✓ should decrease the wallet's pending redemptions value
                        ✓ should decrease Bridge's balance in Bank
                        ✓ should not transfer anything to the treasury
                        ✓ should not change redeemer balance in any way
                      when the single output is a reported timed out requested redemption
                        ✓ should remove the timed out request from the contract state
                        ✓ should delete the wallet's main UTXO
                        ✓ should mark the previous main UTXO as spent
                        ✓ should not change the wallet's pending redemptions value
                        ✓ should not change Bridge's balance in Bank
                        ✓ should not transfer anything to the treasury
                        ✓ should not change redeemer balance in any way
                      when the single output is a pending requested redemption but redeemed amount is
wrong
                        ✓ should revert
                      when the single output is a reported timed out requested redemption but amount is
wrong
                        ✓ should revert
                      when the single output is a legal P2PKH change with a non-zero value
                        ✓ should revert
                      when the single output is a legal P2WPKH change with a non-zero value
                        ✓ should revert
                      when the single output is an illegal P2SH change with a non-zero value
                        ✓ should revert
                      when the single output is a change with a zero as value
                        ✓ should revert
(node:4571) PromiseRejectionHandledWarning: Promise rejection was handled asynchronously (rejection id:
14)
                      when the single output is a non-requested redemption to an arbitrary script
                        ✓ should revert
                      when the single output is provably unspendable OP_RETURN
                        ✓ should revert
                    when there are multiple outputs
                      when output vector consists only of pending requested redemptions
                        ✓ should close processed redemption requests
                        ✓ should delete the wallet's main UTXO
                        ✓ should mark the previous main UTXO as spent
                        ✓ should decrease the wallet's pending redemptions value
                        ✓ should decrease Bridge's balance in Bank
                        ✓ should not transfer anything to the treasury
                        ✓ should not change redeemers balances in any way
                      when output vector consists of pending requested redemptions and a non-zero change
                        ✓ should close processed redemption requests
                        ✓ should update the wallet's main UTXO
                        ✓ should mark the previous main UTXO as spent
                        ✓ should decrease the wallet's pending redemptions value
                        ✓ should decrease Bridge's balance in Bank
                        ✓ should transfer collected treasury fee
                        ✓ should not change redeemers balances in any way
                      when output vector consists only of reported timed out requested redemptions
                        ✓ should remove the timed out requests from the contract state
                        ✓ should delete the wallet's main UTXO
                        ✓ should mark the previous main UTXO as spent
                        ✓ should not change the wallet's pending redemptions value
                        ✓ should not change Bridge's balance in Bank
                        ✓ should not transfer anything to the treasury
                        ✓ should not change redeemers balances in any way
                      when output vector consists of reported timed out requested redemptions and a non-
zero change
                        ✓ should remove the timed out requests from the contract state
                        ✓ should update the wallet's main UTXO
                        ✓ should mark the previous main UTXO as spent
                        ✓ should not change the wallet's pending redemptions value
                        ✓ should not change Bridge's balance in Bank
                        ✓ should not transfer anything to the treasury
                        ✓ should not change redeemers balances in any way
                      when output vector consists of pending requested redemptions and reported timed out
requested redemptions
                        ✓ should remove the timed out requests from the contract state
                        ✓ should close processed redemption requests
                        ✓ should delete the wallet's main UTXO
                        ✓ should mark the previous main UTXO as spent
                        ✓ should decrease the wallet's pending redemptions value
                        ✓ should decrease Bridge's balance in Bank
                        ✓ should not transfer anything to the treasury
                        ✓ should not change redeemers balances in any way
                      when output vector consists of pending requested redemptions, reported timed out
requested redemptions and a non-zero change
                        ✓ should remove the timed out requests from the contract state
                        ✓ should close processed redemption requests
                        ✓ should update the wallet's main UTXO
                        ✓ should mark the previous main UTXO as spent
                        ✓ should decrease the wallet's pending redemptions value
                        ✓ should decrease Bridge's balance in Bank
                        ✓ should transfer collected treasury fee
                        ✓ should not change redeemers balances in any way
                      when output vector contains a pending requested redemption with wrong amount
redeemed
                        ✓ should revert
                      when output vector contains a reported timed out requested redemption with wrong
amount redeemed
                        ✓ should revert
                      when output vector contains a non-zero P2SH change output
                        ✓ should revert
```

```
                  when output vector contains multiple non-zero change outputs
                    ✓ should revert
                  when output vector contains one change but with zero as value
                    ✓ should revert
                  when output vector contains a non-requested redemption to an arbitrary script hash
                    ✓ should revert
                  when output vector contains a provably unspendable OP_RETURN output
                    ✓ should revert
                when the total transaction fee is too high
                  ✓ should revert
              when wallet state is MovingFunds
                ✓ should succeed
              when wallet state is neither Live nor MovingFunds
                when wallet state is Unknown
                  ✓ should revert
                when wallet state is Closing
                  ✓ should revert
                when wallet state is Closed
                  ✓ should revert
                when wallet state is Terminated
                  ✓ should revert
            when the single input doesn't point to the wallet's main UTXO
              ✓ should revert
          when input count is other than one
            ✓ should revert
        when main UTXO data are invalid
          ✓ should revert
      when there is no main UTXO for the given wallet
        ✓ should revert
    when transaction proof is not valid
      when input vector is not valid
        ✓ should revert
      when output vector is not valid
        ✓ should revert
      when transaction is not on same level of merkle tree as coinbase
        ✓ should revert
      when merkle proof is not valid
        ✓ should revert
      when coinbase merkle proof is not valid
        ✓ should revert
      when proof difficulty is not current nor previous
        ✓ should revert
      when headers chain length is not valid
        ✓ should revert
      when headers chain is not valid
        ✓ should revert
      when the work in the header is insufficient
        ✓ should revert
      when accumulated difficulty in headers chain is insufficient
Warning: Potentially unsafe deployment of BridgeStub

    You are using the `unsafeAllow.external-library-linking` flag to include external libraries.
    Make sure you have manually checked that the linked libraries are upgrade safe.

        ✓ should revert
      when transaction data is limited to 64 bytes
        ✓ should revert
  notifyRedemptionTimeout
    when redemption request exists
      when the redemption request has timed out
        when the wallet is in Live state
          when the wallet is the active wallet
            ✓ should update the wallet's pending redemptions value
            ✓ should return the requested amount of tokens to the redeemer
            ✓ should remove the request from the pending redemptions
            ✓ should add the request to the timed-out redemptions
            ✓ should change the wallet's state to MovingFunds
            ✓ should set the wallet's move funds requested timestamp
            ✓ should emit WalletMovingFunds event
            ✓ should delete the active wallet public key hash
            ✓ should call the ECDSA wallet registry's seize function
            ✓ should emit RedemptionTimedOut event
```

```
              ✓ should decrease the live wallets counter
            when the wallet is not the active wallet
              ✓ should not delete the active wallet public key hash
          when the wallet is in MovingFunds state
            ✓ should update the wallet's pending redemptions value
            ✓ should return the requested amount of tokens to the redeemer
            ✓ should remove the request from the pending redemptions
            ✓ should add the request to the timed-out redemptions
            ✓ should not change wallet state
            ✓ should call the ECDSA wallet registry's seize function
            ✓ should emit RedemptionTimedOut event
          when the wallet is in Terminated state
            ✓ should update the wallet's pending redemptions value
            ✓ should remove the request from the pending redemptions
            ✓ should add the request to the timed-out redemptions
            ✓ should not change wallet state
            ✓ should emit RedemptionTimedOut event
            ✓ should return the requested amount of tokens to the redeemer
            ✓ should not call the ECDSA wallet registry's seize function
          when the wallet is neither in Live, MovingFunds nor Terminated state
            when wallet state is Unknown
              ✓ should revert
            when wallet state is Closing
              ✓ should revert
            when wallet state is Closed
              ✓ should revert
        when the redemption request has not timed out
          ✓ should revert
      when redemption request does not exist
        ✓ should revert
    notifyRedemptionVeto
      when the caller is not the redemption watchtower
        ✓ should revert
      when the caller is the redemption watchtower
        when the redemption does not exist
          ✓ should revert
        when the redemption exists
          ✓ should update the wallet's pending redemptions value
          ✓ should remove the request from the pending redemptions
          ✓ should transfer the requested amount of tokens to the watchtower

  Bridge - Vaults
    isVaultTrusted
      ✓ should not trust a vault by default
    setVaultStatus
      when called not by the governance
        ✓ should revert
        ✓ should revert
      when called by the governance
        when setting vault status as trusted
          ✓ should correctly update vault status
          ✓ should emit VaultStatusUpdated event
        when setting vault status as no longer trusted
          ✓ should correctly update vault status
          ✓ should emit VaultStatusUpdated event

  Bridge - Wallets
    requestNewWallet
      when called by a third party
        when wallet creation is not in progress
          when active wallet is not set
            ✓ should emit NewWalletRequested event
            ✓ should call ECDSA Wallet Registry's requestNewWallet function
          when active wallet is set
            when active wallet has a main UTXO set
              when the active wallet main UTXO data are valid
                when wallet creation conditions are met
                  when active wallet is old enough and its balance is greater or equal the minimum BTC
balance threshold
                    ✓ should emit NewWalletRequested event
                    ✓ should call ECDSA Wallet Registry's requestNewWallet function
                  when active wallet is not old enough but its balance is greater or equal the maximum
```

```
BTC balance threshold
                    ✓ should emit NewWalletRequested event
                    ✓ should call ECDSA Wallet Registry's requestNewWallet function
                when active wallet is not old enough and its balance is greater or equal the minimum but
lesser than the maximum BTC balance threshold
                  ✓ should revert
(node:4571) PromiseRejectionHandledWarning: Promise rejection was handled asynchronously (rejection id:
16)
                when active wallet is old enough but its balance is lesser than the minimum BTC balance
threshold
                  ✓ should revert
(node:4571) PromiseRejectionHandledWarning: Promise rejection was handled asynchronously (rejection id:
17)
              when the active wallet main UTXO data are invalid
                ✓ should revert
            when active wallet has no main UTXO set
              when the minimum BTC balance threshold is non-zero
                ✓ should revert
              when the minimum BTC balance threshold is non-zero
                ✓ should revert
              when the minimum BTC balance threshold is zero
                when wallet creation conditions are met
                  ✓ should emit NewWalletRequested event
                  ✓ should call ECDSA Wallet Registry's requestNewWallet function
        when wallet creation is already in progress
          when wallet creation state is AWAITING_SEED
            ✓ should revert
          when wallet creation state is AWAITING_RESULT
            ✓ should revert
          when wallet creation state is CHALLENGE
            ✓ should revert
    __ecdsaWalletCreatedCallback
      when called by a third party
        ✓ should revert
      when called by the ECDSA Wallet Registry
        when called with a valid ECDSA Wallet details
          ✓ should register ECDSA wallet reference
          ✓ should transition wallet to Live state
          ✓ should set the created at timestamp
          ✓ should set the wallet as the active one
          ✓ should emit NewWalletRegistered event
          ✓ should increase the live wallets counter
        when called with the ECDSA Wallet already registered
          with unique wallet ID and unique public key
            ✓ should not revert
          with duplicated wallet ID and unique public key
            ✓ should not revert
          with unique wallet ID, unique public key X and duplicated public key Y
            ✓ should not revert
          with unique wallet ID, unique public key Y and duplicated public key X
            ✓ should not revert
          with unique wallet ID and duplicated public key
            ✓ should revert
          with duplicated wallet ID and duplicated public key
            ✓ should revert
    __ecdsaWalletHeartbeatFailedCallback
      when called by the ECDSA Wallet Registry
        when wallet is in Live state
          when wallet balance is zero
            when wallet is the active one
              ✓ should change wallet's state to Closing
              ✓ should set the wallet's closing started timestamp
              ✓ should emit WalletClosing event
              ✓ should unset the active wallet
              ✓ should decrease the live wallets counter
            when wallet is not the active one
              ✓ should change wallet's state to Closing
              ✓ should set the wallet's closing started timestamp
              ✓ should emit WalletClosing event
              ✓ should not unset the active wallet
              ✓ should decrease the live wallets counter
          when wallet balance is greater than zero
```

```
            when wallet is the active one
              ✓ should change wallet's state to MovingFunds
              ✓ should set move funds requested at timestamp
              ✓ should emit WalletMovingFunds event
              ✓ should unset the active wallet
              ✓ should decrease the live wallets counter
            when wallet is not the active one
              ✓ should change wallet's state to MovingFunds
              ✓ should set move funds requested at timestamp
              ✓ should emit WalletMovingFunds event
              ✓ should not unset the active wallet
              ✓ should decrease the live wallets counter
        when wallet is not in Live state
          when wallet state is Unknown
            ✓ should revert
          when wallet state is MovingFunds
            ✓ should revert
          when wallet state is Closing
            ✓ should revert
          when wallet state is Closed
            ✓ should revert
          when wallet state is Terminated
            ✓ should revert
      when called by a third party
        ✓ should revert
    notifyWalletCloseable
      when the reported wallet is not the active one
        when wallet is in Live state
          when wallet reached the maximum age
            when wallet balance is zero
              ✓ should change wallet's state to Closing
              ✓ should set the wallet's closing started timestamp
              ✓ should emit WalletClosing event
              ✓ should decrease the live wallets counter
            when wallet balance is greater than zero
              ✓ should change wallet's state to MovingFunds
              ✓ should set move funds requested at timestamp
              ✓ should emit WalletMovingFunds event
              ✓ should decrease the live wallets counter
          when wallet did not reach the maximum age but their balance is lesser than the minimum
threshold
            when wallet balance is zero
              ✓ should change wallet's state to Closing
              ✓ should set the wallet's closing started timestamp
              ✓ should emit WalletClosing event
              ✓ should decrease the live wallets counter
            when wallet balance is greater than zero
              ✓ should change wallet's state to MovingFunds
              ✓ should set move funds requested at timestamp
              ✓ should emit WalletMovingFunds event
              ✓ should decrease the live wallets counter
          when wallet did not reach the maximum age and their balance is greater or equal the minimum
threshold
            ✓ should revert
          when wallet did not reach the maximum age and invalid main UTXO data is passed
            ✓ should revert
        when wallet is not in Live state
          when wallet state is Unknown
            ✓ should revert
          when wallet state is MovingFunds
            ✓ should revert
          when wallet state is Closing
            ✓ should revert
          when wallet state is Closed
            ✓ should revert
          when wallet state is Terminated
            ✓ should revert
      when the reported wallet is the active one
        ✓ should revert
    notifyWalletClosingPeriodElapsed
      when the wallet is in the Closing state
        when closing period has elapsed
          ✓ should set wallet state to Closed
          ✓ should emit WalletClosed event
          ✓ should call the ECDSA wallet registry's closeWallet function
        when closing period has not elapsed yet
          ✓ should revert
      when the wallet is not in the Closing state
        when wallet state is Unknown
          ✓ should revert
        when wallet state is Live
          ✓ should revert
        when wallet state is MovingFunds
          ✓ should revert
        when wallet state is Closed
          ✓ should revert
        when wallet state is Terminated
          ✓ should revert

  Deployment
    Bridge
      ✓ should set Bridge proxy admin
      ✓ should set ProxyAdmin owner
      ✓ should set Bridge implementation
      ✓ should set Bridge implementation in ProxyAdmin
      ✓ should set implementation address different than proxy address
      ✓ should set Bridge governance
      ✓ should revert when initialize called again
    BridgeGovernance
      ✓ should set owner
    WalletRegistry
      ✓ should set walletOwner
    Bank
      ✓ should set Bridge reference
      ✓ should set Bank owner
    TBTCVault
      ✓ should set Bank reference
      ✓ should set TBTC reference
      ✓ should set TBTCVault owner
    MaintainerProxy
      ✓ should set Bridge reference
      ✓ should set ReimbursementPool reference
      ✓ should set MaintainerProxy owner
    ReimbursementPool
      ✓ should authorize MaintainerProxy in ReimbursementPool
      ✓ should set ReimbursementPool owner
    VendingMachine
      ✓ should set vendingMachineUpgradeInitiator
      ✓ should set unmintFeeUpdateInitiator
      ✓ should set VendingMachine owner

  EcdsaLib
    compressPublicKey
      with valid uncompressed public key
        ✓ with even Y
        ✓ with odd Y
        ✓ with leading zeros
        ✓ with trailing zeros

  Heartbeat
    when the message is empty
      ✓ should return false
    when the message has less than 16 bytes
      ✓ should return false
    when the message has more than 16 bytes
      ✓ should return false
    when the message has 16 bytes
      when the message does not have the required prefix
        ✓ should return false
      when the message has the required prefix
        ✓ should return true

  RedemptionWatchtower
    enableWatchtower
```

```
    when called not by the owner
      ✓ should revert
    when called by the owner
      when already enabled
        ✓ should revert
      when not enabled yet
        when manager address is zero
          ✓ should revert
        when manager address is non-zero
          ✓ should set the enabledAt timeout properly
          ✓ should set the watchtower manager properly
          ✓ should set initial guardians properly
          ✓ should emit WatchtowerEnabled event
          ✓ should emit GuardianAdded events
  disableWatchtower
    when the watchtower is not enabled
      ✓ should revert
    when the watchtower is enabled
      when the watchtower is disabled already
        ✓ should revert
      when the watchtower is not disabled yet
        when the watchtower lifetime is not expired
          ✓ should revert
        when the watchtower lifetime is expired
          ✓ should set the disabledAt timeout properly
          ✓ should emit WatchtowerDisabled event
  addGuardian
    when watchtower manager is not set
      ✓ should revert
    when watchtower manager is set
      when called not by the watchtower manager
        ✓ should revert
      when called by the watchtower manager
        when guardian already exists
          ✓ should revert
        when guardian does not exist
          ✓ should add the guardian properly
          ✓ should emit GuardianAdded event
  removeGuardian
    when called not by the governance
      ✓ should revert
    when called by the governance
      when guardian does not exist
        ✓ should revert
      when guardian exists
        ✓ should remove the guardian properly
        ✓ should emit GuardianRemoved event
  raiseObjection
    when called not by a guardian
      ✓ should revert
    when called by a guardian
      when redemption request is already vetoed
        ✓ should revert
      when redemption request is not vetoed yet
        when guardian already objected
          ✓ should revert
        when guardian did not object yet
          when redemption request does not exist
            ✓ should revert
          when redemption request exists
            when the requested amount is below the waived amount limit
              ✓ should revert
            when watchtower has been disabled
              ✓ should revert
            when delay period expired and request was created after mechanism initialization
              when the raised objection is the first one
                ✓ should revert
              when the raised objection is the second one
                ✓ should revert
              when the raised objection is the third one
                ✓ should revert
            when delay period expired but request was created before mechanism initialization
```

```
              when the raised objection is the first one
                ✓ should emit VetoPeriodCheckOmitted event
                ✓ should store the objection key
                ✓ should update veto state properly
                ✓ should emit ObjectionRaised event
              when the raised objection is the second one
                ✓ should emit VetoPeriodCheckOmitted event
                ✓ should store the objection key
                ✓ should update veto state properly
                ✓ should emit ObjectionRaised event
              when the raised objection is the third one
                ✓ should emit VetoPeriodCheckOmitted event
                ✓ should store the objection key
                ✓ should update veto state properly
                ✓ should emit ObjectionRaised event
                ✓ should mark the redeemer as banned
                ✓ should emit Banned event
                ✓ should emit VetoFinalized event
                ✓ should decrease wallet's pending redemptions value in the Bridge
                ✓ should remove pending redemption in the Bridge
                ✓ should transfer the redemption amount from the Bridge
                ✓ should leave a proper withdrawable amount and burn the penalty fee
            when delay period did not expire yet
              when the raised objection is the first one
                ✓ should not emit VetoPeriodCheckOmitted event
                ✓ should store the objection key
                ✓ should update veto state properly
                ✓ should emit ObjectionRaised event
              when the raised objection is the second one
                ✓ should not emit VetoPeriodCheckOmitted event
                ✓ should store the objection key
                ✓ should update veto state properly
                ✓ should emit ObjectionRaised event
              when the raised objection is the third one
                ✓ should not emit VetoPeriodCheckOmitted event
                ✓ should store the objection key
                ✓ should update veto state properly
                ✓ should emit ObjectionRaised event
                ✓ should mark the redeemer as banned
                ✓ should emit Banned event
                ✓ should emit VetoFinalized event
                ✓ should decrease wallet's pending redemptions value in the Bridge
                ✓ should remove pending redemption in the Bridge
                ✓ should transfer the redemption amount from the Bridge
                ✓ should leave a proper withdrawable amount and burn the penalty fee
      getRedemptionDelay
        when the redemption request does not exist
          ✓ should revert
        when the redemption request exists
          when the watchtower has been disabled
            ✓ should return zero as the delay
          when the watchtower has not been disabled
            when the requested amount is below the waived limit
              ✓ should return zero as the delay
            when the requested amount is not below the waived limit
              when there are no objections
                ✓ should return the default delay
              when there is one objection
                ✓ should return the level-one delay
              when there are two objections
                ✓ should return the level-two delay
              when there are three objections
                ✓ should revert
      updateWatchtowerParameters
        when called not by the watchtower manager
          ✓ should revert
        when called by the watchtower manager
          when new parameters are invalid
            when the new lifetime is lesser than the current one
              ✓ should revert
            when the new veto penalty fee is not in the proper range
              ✓ should revert
        when level-two delay is lesser than level-one delay
          ✓ should revert
        when level-one delay is lesser than default delay
          ✓ should revert
      when all new parameters are valid
        when watchtower lifetime is increased
          ✓ should emit WatchtowerParametersUpdated event
          ✓ should update the watchtower parameters
        when veto penalty is changed to to the maximum value of 5%
          ✓ should emit WatchtowerParametersUpdated event
          ✓ should update the watchtower parameters
        when veto penalty is changed to to the middle of the range
          ✓ should emit WatchtowerParametersUpdated event
          ✓ should update the watchtower parameters
        when veto penalty is changed to the minimum value of 0%
          ✓ should emit WatchtowerParametersUpdated event
          ✓ should update the watchtower parameters
        when veto freeze period is changed to a non-zero value
          ✓ should emit WatchtowerParametersUpdated event
          ✓ should update the watchtower parameters
        when veto freeze period is changed to 0
          ✓ should emit WatchtowerParametersUpdated event
          ✓ should update the watchtower parameters
        when delays are changed to a non-zero value
          ✓ should emit WatchtowerParametersUpdated event
          ✓ should update the watchtower parameters
        when delays are changed to 0
          ✓ should emit WatchtowerParametersUpdated event
          ✓ should update the watchtower parameters
        when waived amount limit is changed to a non-zero value
          ✓ should emit WatchtowerParametersUpdated event
          ✓ should update the watchtower parameters
  isSafeRedemption
    when the balance owner is banned
      ✓ should return false
    when the redeemer is banned
      ✓ should return false
    when redemption key was vetoed
      ✓ should return false
    when redemption key was objected but not vetoed
      ✓ should return false
    when all safety criteria are met
      ✓ should return true
  unban
    when the caller is not the watchtower manager
      ✓ should revert
    when the caller is the watchtower manager
      when the redeemer is not banned
        ✓ should revert
      when the redeemer is banned
        ✓ should remove the redeemer from the banned list
        ✓ should emit Unbanned event
  withdrawVetoedFunds
    when the veto is not finalized
      when there are no objections at all
        ✓ should revert
      when there some objections
        ✓ should revert
    when the veto is finalized and the penalty fee is lesser than 100%
      when the caller is not the redeemer
        ✓ should revert
      when the caller is the redeemer
        when the freeze period has not expired
          ✓ should revert
        when the freeze period has expired
          when there are no funds to withdraw
            ✓ should revert
          when there are funds to withdraw
            ✓ should emit VetoedFundsWithdrawn event
            ✓ should set withdrawable amount to zero
            ✓ should transfer the funds to the redeemer
    when the veto is finalized and the penalty fee is 100%
```

```
        when the caller is not the redeemer
          ✓ should revert
        when the caller is the redeemer
          when the freeze period has not expired
            ✓ should revert
          when the freeze period has expired
            ✓ should revert

  VendingMachine - Upgrade
    upgrade process - option #1
      step#1 - TBTC v1 transfer
        ✓ should transfer all TBTC v1 to TBTCVault
      step#2 - TBTC v1 withdrawal
        ✓ should let the governance withdraw TBTC v1 from TBTCVault
      step#3 - BTC deposit
        ✓ should let the governance donate TBTCVault
      step#4 - functioning system
        ✓ should let TBTC v2 holders unmint their tokens
        ✓ should let Bank balance holders mint TBTC v2
    upgrade process - option #2
      step#1 - TBTC v1 transfer
        ✓ should transfer all TBTC v1 to TBTCVault
      step#2 - TBTC v1 transfer back to VendingMachine
        ✓ should let the governance transfer TBTC v1 back to VendingMachine
      step #3 - BTC deposit
        ✓ should let to deposit BTC into v2 Bridge
        step #4 - TBTC v2 -> v2 unminting
          ✓ should let the redeemer to unmint TBTC v2 back to TBTC v1

  VendingMachine
    mint
      when TBTC v1 owner has not enough tokens
        ✓ should revert
      when TBTC v1 owner has enough tokens
        when minting entire allowance
          ✓ should mint the same amount of TBTC v2
          ✓ should transfer TBTC v1 tokens to the VendingMachine
          ✓ should emit Minted event
        when minting part of the allowance
          ✓ should mint the same amount of TBTC v2
          ✓ should transfer TBTC v1 tokens to the VendingMachine
          ✓ should emit Minted event
    receiveApproval
      when called directly
        ✓ should revert
      when called not for TBTC v1 token
        ✓ should revert
      when called via approveAndCall
        ✓ should mint TBTC v2 to the caller
        ✓ should transfer TBTC v1 tokens to the VendingMachine
        ✓ should emit Minted event
    unmint
      when unmint fee is zero
        when TBTC v2 owner has not enough tokens
          ✓ should revert
        when TBTC v2 owner has enough tokens
          when unminting entire TBTC v2 balance
            ✓ should transfer no TBTC v2 to the VendingMachine
            ✓ should burn unminted TBTC v2 tokens
            ✓ should transfer unminted TBTC v1 tokens back to the owner
            ✓ should emit the Unminted event
          when unminting part of TBTC v2 balance
            ✓ should transfer no TBTC v2 to the VendingMachine
            ✓ should burn unminted TBTC v2 tokens
            ✓ should transfer unminted TBTC v1 tokens back to the owner
            ✓ should emit the Unminted event
      when unmint fee is non-zero
        when TBTC v2 owner has not enough tokens
          ✓ should revert
        when TBTC v2 owner has enough tokens
          when unminting entire TBTC v2 balance
            ✓ should transfer TBTC v2 fee to the VendingMachine
          ✓ should burn unminted TBTC v2 tokens
          ✓ should transfer unminted TBTC v1 tokens back to the owner
          ✓ should emit the Unminted event
        when unminting part of TBTC v2 balance
          ✓ should transfer TBTC v2 fee to the VendingMachine
          ✓ should burn unminted TBTC v2 tokens
          ✓ should transfer unminted TBTC v1 tokens back to the owner
          ✓ should emit the Unminted event
    withdrawFees
      when caller is not the owner
        ✓ should revert
      when caller is the owner
        ✓ should withdraw the provided amount of fees
        ✓ should leave the rest of fees in VendingMachine
    initiateUnmintFeeUpdate
      when caller is a third party
        ✓ should revert
      when caller is the contract owner
        ✓ should revert
      when caller is the update initiator
        ✓ should not update the unmint fee
        ✓ should start the update initiation time
        ✓ should set the pending new unmint fee
        ✓ should start the governance delay timer
        ✓ should emit UnmintFeeUpdateInitiated event
    finalizeUnmintFeeUpdate
      when caller is a third party
        ✓ should revert
      when caller is the update initiator
        ✓ should revert
      when caller is the owner
        when update process is not initialized
          ✓ should revert
        when update process is initialized
          when governance delay has not passed
            ✓ should revert
          when governance delay passed
            ✓ should update the unmint fee
            ✓ should emit UnmintFeeUpdated event
            ✓ should reset the governance delay timer
            ✓ should reset the pending new unmint fee
            ✓ should reset the unmint fee update initiated timestamp
    initiateVendingMachineUpgrade
      when caller is a third party
        ✓ should revert
      when caller is the contract owner
        ✓ should revert
      when caller is the upgrade initiator
        when new vending machine address is zero
          ✓ should revert
        when new vending machine address is non-zero
          ✓ should not transfer token ownership
          ✓ should start the upgrade initiation time
          ✓ should set the pending new vending machine address
          ✓ should start the governance delay timer
          ✓ should emit VendingMachineUpgradeInitiated event
    finalizeVendingMachineUpgrade
      when caller is a third party
        ✓ should revert
      when caller is the upgrade initiator
        ✓ should revert
      when caller is the owner
        when upgrade process is not initialized
          ✓ should revert
        when upgrade process is initialized
          when governance delay has not passed
            ✓ should revert
          when governance delay passed
            ✓ should transfer token ownership to the new VendingMachine
            ✓ should transfer all TBTC v1 to the new VendingMachine
            ✓ should emit VendingMachineUpgraded event
            ✓ should reset the governance delay timer
          ✓ should reset the pending new vending machine address
          ✓ should reset the vending machine update initiated timestamp
    transferUnmintFeeUpdateInitiatorRole
      when caller is the owner
        ✓ should revert
      when caller is a third party
        ✓ should revert
      when caller is the update initiator
        when new initiator is a valid address
          ✓ should transfer the role
        when new initiator is zero address
          ✓ should revert
    transferVendingMachineUpgradeInitiatorRole
      when caller is the owner
        ✓ should revert
      when caller is a third party
        ✓ should revert
      when caller is the update initiator
        when new initiator is a valid address
          ✓ should transfer the role
        when new initiator is zero address
          ✓ should revert
    unmintFeeFor
      when unmint fee is non-zero
        ✓ should return a correct portion of the amount to unmint
      when unmint fee is zero
        ✓ should return zero

  VendingMachineV2
    exchange
      when tBTC v1 exchanger has not enough tokens
        ✓ should revert
      when not enough tBTC v2 was deposited
        ✓ should revert
      when exchanging entire allowance
        ✓ should exchange the same amount of tBTC v2
        ✓ should transfer tBTC v1 tokens to the VendingMachineV2
        ✓ should emit Exchanged event
      when exchanging part of the allowance
        ✓ should exchange the same amount of tBTC v2
        ✓ should transfer tBTC v1 tokens to the VendingMachineV2
        ✓ should emit Exchanged event
    receiveApproval
      when called directly
        ✓ should revert
      when called not for tBTC v1 token
        ✓ should revert
      when called via approveAndCall
        ✓ should exchange tBTC v2 with the caller
        ✓ should transfer tBTC v1 tokens to the VendingMachineV2
        ✓ should emit Exchanged event
    depositTBTCV2
      when depositing entire allowance
        ✓ should transfer tBTC v2 to the VendingMachineV2
        ✓ should emit Deposited event
      when depositing part of the allowance
        ✓ should transfer tBTC v2 to the VendingMachineV2
        ✓ should emit Deposited event
    withdrawFunds
      when called by third party
        ✓ should revert
      when called by the owner
        when withdrawing tBTC v1 tokens
          ✓ should transfer tokens to the recipient
          ✓ should emit Withdrawn event
        when withdrawing tBTC v2 tokens
          ✓ should transfer tokens to the recipient
          ✓ should emit Withdrawn event

  VendingMachineV3
    exchange
      when tBTC v1 exchanger has not enough tokens
```

```
        ✓ should revert
      when not enough tBTC v2 was deposited
        ✓ should revert
      when exchanging entire allowance
        ✓ should exchange the same amount of tBTC v2
        ✓ should transfer tBTC v1 tokens to the VendingMachineV3
        ✓ should emit Exchanged event
      when exchanging part of the allowance
        ✓ should exchange the same amount of tBTC v2
        ✓ should transfer tBTC v1 tokens to the VendingMachineV3
        ✓ should emit Exchanged event
    receiveApproval
      when called directly
        ✓ should revert
      when called not for tBTC v1 token
        ✓ should revert
      when called via approveAndCall
        ✓ should exchange tBTC v2 with the caller
        ✓ should transfer tBTC v1 tokens to the VendingMachineV3
        ✓ should emit Exchanged event
    depositTBTCV2
      when depositing entire allowance
        ✓ should transfer tBTC v2 to the VendingMachineV3
        ✓ should emit Deposited event
      when depositing part of the allowance
        ✓ should transfer tBTC v2 to the VendingMachineV3
        ✓ should emit Deposited event
    recoverFunds
      when called by third party
        ✓ should revert
      when called by the owner
        when recovering tBTC v1 tokens
          ✓ should transfer tokens to the recipient
          ✓ should emit FundsRecovered event
        when recovering tBTC v2 tokens
          ✓ should revert
        when recovering other tokens
          ✓ should transfer tokens to the recipient
          ✓ should emit FundsRecovered event
    withdrawTbtcV2
      when called by a third party
        ✓ should revert
      when called by the owner
        when some tBTC v1 would be unbacked
          ✓ should revert
        when all tBTC v1 would be still backed
          ✓ should transfer tokens to the recipient
          ✓ should emit TbtcV2Withdrawn event

  WalletProposalValidator
    validateDepositSweepProposal
      when wallet is incorrect state
        when wallet state is Unknown
          ✓ should revert
        when wallet state is Closing
          ✓ should revert
        when wallet state is Closed
          ✓ should revert
        when wallet state is Terminated
          ✓ should revert
      when wallet is correct state
        when wallet state is Live
          when sweep is below the min size
            ✓ should revert
          when sweep is above the min size
            when sweep exceeds the max size
              ✓ should revert
            when sweep does not exceed the max size
              when deposit extra info length does not match
                ✓ should revert
              when deposit extra info length matches
                when proposed sweep tx fee is invalid
          when proposed sweep tx fee is zero
            ✓ should revert
          when proposed sweep tx fee is greater than the allowed
            ✓ should revert
        when proposed sweep tx fee is valid
          when there is a non-revealed deposit
            ✓ should revert
          when all deposits are revealed
            when there is an immature deposit
              ✓ should revert
            when all deposits achieved the min age
              when there is an already swept deposit
                ✓ should revert
              when all deposits are not swept yet
                when there is a deposit with invalid extra info
                  when funding tx hashes don't match
                    ✓ should revert
                  when 20-byte funding output hash does not match
                    ✓ should revert
                  when 32-byte funding output hash does not match
                    ✓ should revert
                when all deposits extra info are valid
                  when there is a deposit that violates the refund safety margin
                    ✓ should revert
                  when all deposits preserve the refund safety margin
                    when there is a deposit controlled by a different wallet
                      ✓ should revert
                    when all deposits are controlled by the same wallet
                      when there is a deposit targeting a different vault
                        ✓ should revert
                      when all deposits targets the same vault
                        when there are duplicated deposits
                          ✓ should revert
                        when all deposits are unique
                          ✓ should succeed
  when wallet state is MovingFunds
    when sweep is below the min size
      ✓ should revert
    when sweep is above the min size
      when sweep exceeds the max size
        ✓ should revert
      when sweep does not exceed the max size
        when deposit extra info length does not match
          ✓ should revert
        when deposit extra info length matches
          when proposed sweep tx fee is invalid
            when proposed sweep tx fee is zero
              ✓ should revert
            when proposed sweep tx fee is greater than the allowed
              ✓ should revert
          when proposed sweep tx fee is valid
            when there is a non-revealed deposit
              ✓ should revert
            when all deposits are revealed
              when there is an immature deposit
                ✓ should revert
              when all deposits achieved the min age
                when there is an already swept deposit
                  ✓ should revert
                when all deposits are not swept yet
                  when there is a deposit with invalid extra info
                    when funding tx hashes don't match
                      ✓ should revert
                    when 20-byte funding output hash does not match
                      ✓ should revert
                    when 32-byte funding output hash does not match
                      ✓ should revert
                  when all deposits extra info are valid
                    when there is a deposit that violates the refund safety margin
                      ✓ should revert
                    when all deposits preserve the refund safety margin
                      when there is a deposit controlled by a different wallet
```

```
                              ✓ should revert
                            when all deposits are controlled by the same wallet
                              when there is a deposit targeting a different vault
                                ✓ should revert
                              when all deposits targets the same vault
                                when there are duplicated deposits
                                  ✓ should revert
                                when all deposits are unique
                                  ✓ should succeed
      validateRedemptionProposal
        when wallet is in incorrect state
          when wallet state is Unknown
            ✓ should revert
          when wallet state is Closing
            ✓ should revert
          when wallet state is Closed
            ✓ should revert
          when wallet state is Terminated
            ✓ should revert
        when wallet is in correct state
          when wallet state is Live
            when redemption is below the min size
              ✓ should revert
            when redemption is above the min size
              when redemption exceeds the max size
                ✓ should revert
              when redemption does not exceed the max size
                when proposed redemption tx fee is invalid
                  when proposed redemption tx fee is zero
                    ✓ should revert
                  when proposed redemption tx fee is greater than the allowed total fee
                    ✓ should revert
                when proposed redemption tx fee is valid
                  when there is a non-pending request
                    ✓ should revert
                  when all requests are pending
                    when there is an immature request
                      when immaturity is caused by REDEMPTION_REQUEST_MIN_AGE violation
                        ✓ should revert
                      when immaturity is caused by watchtower's delay violation
                        ✓ should revert
                    when all requests achieved the min age
                      when there is a request that violates the timeout safety margin
                        ✓ should revert
                      when all requests preserve the timeout safety margin
                        when there is a request that incurs an unacceptable tx fee share
                          when there is no fee remainder
                            ✓ should revert
                          when there is a fee remainder
                            ✓ should revert
                        when all requests incur an acceptable tx fee share
                          when there are duplicated requests
                            ✓ should revert
                          when all requests are unique
                            when watchtower is not set
                              ✓ should succeed
                            when watchtower is set
                              ✓ should succeed
          when wallet state is MovingFunds
            when redemption is below the min size
              ✓ should revert
            when redemption is above the min size
              when redemption exceeds the max size
                ✓ should revert
              when redemption does not exceed the max size
                when proposed redemption tx fee is invalid
                  when proposed redemption tx fee is zero
                    ✓ should revert
                  when proposed redemption tx fee is greater than the allowed total fee
                    ✓ should revert
                when proposed redemption tx fee is valid
                  when there is a non-pending request
              ✓ should revert
          when all requests are pending
            when there is an immature request
              when immaturity is caused by REDEMPTION_REQUEST_MIN_AGE violation
                ✓ should revert
              when immaturity is caused by watchtower's delay violation
                ✓ should revert
            when all requests achieved the min age
              when there is a request that violates the timeout safety margin
                ✓ should revert
              when all requests preserve the timeout safety margin
                when there is a request that incurs an unacceptable tx fee share
                  when there is no fee remainder
                    ✓ should revert
                  when there is a fee remainder
                    ✓ should revert
                when all requests incur an acceptable tx fee share
                  when there are duplicated requests
                    ✓ should revert
                  when all requests are unique
                    when watchtower is not set
                      ✓ should succeed
                    when watchtower is set
                      ✓ should succeed
  validateMovingFundsProposal
    when wallet's state is not MovingFunds
      when wallet state is Unknown
        ✓ should revert
      when wallet state is Live
        ✓ should revert
      when wallet state is Closing
        ✓ should revert
      when wallet state is Closed
        ✓ should revert
      when wallet state is Terminated
        ✓ should revert
    when wallet's state is MovingFunds
      when moving funds commitment has not been submitted
        ✓ should revert
      when moving funds commitment has been submitted
        when commitment hash does not match target wallets
          ✓ should revert
        when commitment hash matches target wallets
          when no main UTXO is passed
            ✓ should revert
          when the passed main UTXO is incorrect
            ✓ should revert
          when the passed main UTXO is correct
            when source wallet BTC balance is below dust threshold
              ✓ should revert
            when source wallet BTC balance is equal to or greater that dust threshold
              when transaction fee is zero
                ✓ should revert
              when transaction fee is too high
                ✓ should revert
              when transaction fee is valid
                ✓ should pass validation
  validateMovedFundsSweepProposal
    when wallet's state is incorrect
      when wallet state is Unknown
        ✓ should revert
      when wallet state is Closing
        ✓ should revert
      when wallet state is Closed
        ✓ should revert
      when wallet state is Terminated
        ✓ should revert
    when wallet's state is correct
      when wallet state is Live
        when moved funds sweep request's state is not Pending
          ✓ should revert
        when moved funds sweep request's state is Pending
```

```
          when moved funds sweep request does not belong to the wallet
            ✓ should revert
          when moved funds sweep request belongs to the wallet
            when transaction fee is zero
              ✓ should revert
            when transaction fee is too high
              ✓ should revert
            when transaction fee is valid
              ✓ should pass validation
      when wallet state is MovingFunds
        when moved funds sweep request's state is not Pending
          ✓ should revert
        when moved funds sweep request's state is Pending
          when moved funds sweep request does not belong to the wallet
            ✓ should revert
          when moved funds sweep request belongs to the wallet
            when transaction fee is zero
              ✓ should revert
            when transaction fee is too high
              ✓ should revert
            when transaction fee is valid
              ✓ should pass validation
  validateHeartbeatProposal
    when message is not valid
      ✓ should revert
    when message is valid
      ✓ should succeed

Integration Test - Full flow
  Check deposit and redemption flow
    when wallet is created
      when a deposit is revealed
        - should create a deposit
      when the deposit sweep proof is submitted
        - should mint TBTC tokens for the depositor
        - should increase the balance of vault in the bank
        - should update the main UTXO of the wallet
      when a redemption is requested
        - should create a pending redemption request
        - should increase the pending redemptions value of the wallet
        - should increase the balance of bridge in the bank
      when the redemption proof is submitted
        - should zero the pending redemptions value of the wallet
        - should zero the balance of bridge in the bank
        - should update the main UTXO of the wallet

Integration Test - Slashing
  notifyFraudChallengeDefeatTimeout
    when wallet is created
      when a fraud is reported
        - should slash wallet members
        - should close the wallet in the wallet registry
        - should terminate the wallet in the bridge
        - should consume around 3 100 000 gas for Bridge.notifyMovingFundsTimeoutTx transaction
  notifyRedemptionTimeout
    when wallet is created
      when a redemption timeout is reported
        - should slash wallet members
        - should not close the wallet in the wallet registry
        - should transition the wallet in the bridge to the MovingFunds state
        - should consume around 3 150 000 gas for Bridge.notifyRedemptionTimeout transaction
  notifyMovingFundsTimeout
    when wallet is created
      when moving funds timeout is reported
        - should slash wallet members
        - should close the wallet in the wallet registry
        - should terminate the wallet in the bridge
        - should consume around 3 100 000 gas for Bridge.notifyMovingFundsTimeoutTx transaction

Integration Test - Wallet Creation
  new wallet creation (happy path)
    - should register a new wallet in the WalletRegistry
```

```
      - should register a new wallet details in the Bridge
      - should register a new wallet as active in the Bridge
      - should consume around 94 000 gas for Bridge.requestNewWallet transaction
      - should consume around 341 000 gas for WalletRegistry.approveDkgResult transaction

  AbstractTBTCDepositor
    _initializeDeposit
      when revealed vault does not match
        ✓ should revert
      when revealed vault matches
        when deposit is rejected by the Bridge
          ✓ should revert
        when deposit is accepted by the Bridge
          ✓ should reveal the deposit to the Bridge
          ✓ should return proper values
    _finalizeDeposit
      when deposit is not initialized
        ✓ should revert
      when deposit is already finalized
        ✓ should not revert
      when deposit is initialized but not finalized yet
        when deposit is not finalized by the Bridge
          ✓ should revert
        when deposit is finalized by the Bridge
          when the deposit is swept
            ✓ should return proper values
          when the deposit is optimistically minted
            ✓ should return proper values
    _calculateTbtcAmount
      when all fees are non-zero
        ✓ should return the correct amount
      when all fees are zero
        ✓ should return the correct amount
      when one of the fees is zero
        when treasury fee is zero
          ✓ should return the correct amount
        when optimistic minting fee is zero
          ✓ should return the correct amount
        when transaction max fee is zero
          ✓ should return the correct amount
    _minDepositAmount
      ✓ returns value in TBTC token precision

  L1BitcoinDepositor
    attachL2BitcoinDepositor
      when the caller is not the owner
        ✓ should revert
      when the caller is the owner
        when the L2BitcoinDepositor is already attached
          ✓ should revert
        when the L2BitcoinDepositor is not attached
          when new L2BitcoinDepositor is zero
            ✓ should revert
          when new L2BitcoinDepositor is non-zero
            ✓ should set the l2BitcoinDepositor address properly
    updateReimbursementPool
      when the caller is not the owner
        ✓ should revert
      when the caller is the owner
        ✓ should set the reimbursementPool address properly
        ✓ should emit ReimbursementPoolUpdated event
    updateL2FinalizeDepositGasLimit
      when the caller is not the owner
        ✓ should revert
      when the caller is the owner
        ✓ should set the gas limit properly
        ✓ should emit L2FinalizeDepositGasLimitUpdated event
    updateGasOffsetParameters
      when the caller is not the owner
        ✓ should revert
      when the caller is the owner
        ✓ should set the gas offset params properly
        ✓ should emit GasOffsetParametersUpdated event
    updateReimbursementAuthorization
      when the caller is not the owner
        ✓ should revert
      when the caller is the owner
        ✓ should set the authorization properly
        ✓ should emit ReimbursementAuthorizationUpdated event
    initializeDeposit
      when the L2 deposit owner is zero
        ✓ should revert
      when the L2 deposit owner is non-zero
        when the requested vault is not TBTCVault
          ✓ should revert
        when the requested vault is TBTCVault
          when the deposit state is wrong
            when the deposit state is Initialized
              ✓ should revert
            when the deposit state is Finalized
              ✓ should revert
          when the deposit state is Unknown
            when the reimbursement pool is not set
              ✓ should reveal the deposit to the Bridge
              ✓ should set the deposit state to Initialized
              ✓ should emit DepositInitialized event
              ✓ should not store the deferred gas reimbursement
            when the reimbursement pool is set and caller is authorized
              ✓ should reveal the deposit to the Bridge
              ✓ should set the deposit state to Initialized
              ✓ should emit DepositInitialized event
              ✓ should store the deferred gas reimbursement
            when the reimbursement pool is set and caller is not authorized
              ✓ should reveal the deposit to the Bridge
              ✓ should set the deposit state to Initialized
              ✓ should emit DepositInitialized event
              ✓ should not store the deferred gas reimbursement
    finalizeDeposit
      when the deposit state is wrong
        when the deposit state is Unknown
          ✓ should revert
        when the deposit state is Finalized
          ✓ should revert
      when the deposit state is Initialized
        when the deposit is not finalized by the Bridge
          ✓ should revert
        when the deposit is finalized by the Bridge
          when normalized amount is too low to bridge
            ✓ should revert
          when normalized amount is not too low to bridge
            when payment for Wormhole Relayer is too low
              ✓ should revert
            when payment for Wormhole Relayer is not too low
              when the reimbursement pool is not set
                ✓ should set the deposit state to Finalized
                ✓ should emit DepositFinalized event
                ✓ should increase TBTC allowance for Wormhole Token Bridge
                ✓ should create a proper Wormhole token transfer
                ✓ should send transfer VAA to L2
                ✓ should not call the reimbursement pool
              when the reimbursement pool is set and caller is authorized
                ✓ should set the deposit state to Finalized
                ✓ should emit DepositFinalized event
                ✓ should increase TBTC allowance for Wormhole Token Bridge
                ✓ should create a proper Wormhole token transfer
                ✓ should send transfer VAA to L2
                ✓ should pay out proper reimbursements
              when the reimbursement pool is set and caller is not authorized
                ✓ should set the deposit state to Finalized
                ✓ should emit DepositFinalized event
                ✓ should increase TBTC allowance for Wormhole Token Bridge
                ✓ should create a proper Wormhole token transfer
                ✓ should send transfer VAA to L2
                ✓ should pay out proper reimbursements
    quoteFinalizeDeposit
      ✓ should return the correct cost

L2BitcoinDepositor
  attachL1BitcoinDepositor
    when the caller is not the owner
      ✓ should revert
    when the caller is the owner
      when the L1BitcoinDepositor is already attached
        ✓ should revert
      when the L1BitcoinDepositor is not attached
        when new L1BitcoinDepositor is zero
          ✓ should revert
        when new L1BitcoinDepositor is non-zero
          ✓ should set the l1BitcoinDepositor address properly
  initializeDeposit
    ✓ should emit DepositInitialized event
  receiveWormholeMessages
    when the caller is not the WormholeRelayer
      ✓ should revert
    when the caller is the WormholeRelayer
      when the source chain is not the expected L1
        ✓ should revert
      when the source chain is the expected L1
        when the source address is not the L1BitcoinDepositor
          ✓ should revert
        when the source address is the L1BitcoinDepositor
          when the number of additional VAAs is not 1
            ✓ should revert
          when the number of additional VAAs is 1
            ✓ should pass the VAA to the L2WormholeGateway

L2TBTC
  ✓ should have a name
  ✓ should have a symbol
  ✓ should have 18 decimals
  addMinter
    when called not by the owner
      ✓ should revert
    when called by the owner
      when address is a new minter
        ✓ should add address as a minter
        ✓ should emit an event
      when address is already a minter
        ✓ should revert
      when there are multiple minters
        ✓ should add them into the list
  removeMinter
    when called not by the owner
      ✓ should revert
    when called by the owner
      when address is not a minter
        ✓ should revert
      when a minter address is removed
        ✓ should take minter role from the address
        ✓ should emit an event
      when there are multiple minters
        when deleting the first minter
          ✓ should update the minters list
        when deleting the last minter
          ✓ should update the minters list
        when deleting minter from the middle of the list
          ✓ should update the minters list
  addGuardian
    when called not by the owner
      ✓ should revert
    when called by the owner
      when address is a new guardian
        ✓ should add address as a guardian
        ✓ should emit an event
      when address is already a guardian
        ✓ should revert
        when there are multiple guardians
          ✓ should add them into the list
    removeGuardian
      when called not by the owner
        ✓ should revert
      when called by the owner
        when address is not a guardian
          ✓ should revert
        when a guardian address is removed
          ✓ should take guardian role from the address
          ✓ should emit an event
        when there are multiple guardians
          when deleting the first guardian
            ✓ should update the guardians list
          when deleting the last guardian
            ✓ should update the guardians list
          when deleting guardian from the middle of the list
            ✓ should update the guardians list
    recoverERC20
      when called not by the owner
        ✓ should revert
      when called by the contract owner
        ✓ should transfer tokens to the recipient
    recoverERC721
      when called not by the owner
        ✓ should revert
      when called by the owner
        ✓ transfers token to the recipient
    pause
      when called not by a guardian
        ✓ should revert
      when called by a guardian
        ✓ should emit Paused event
        ✓ should pause mint functionality
        ✓ should pause burn functionality
        ✓ should pause burnFrom functionality
        ✓ should not pause transfers
    unpause
      when called not by the owner
        ✓ should revert
      when called by the owner
        ✓ should emit Unpaused event
        ✓ should unpause mint functionality
        ✓ should unpause burn functionality
        ✓ should unpause burnFrom functionality
    mint
      when called not by a minter
        ✓ should revert
      when called by a minter
        for a zero account
          ✓ should revert
        for a non-zero account
          ✓ should increment totalSupply
          ✓ should increment recipient balance
          ✓ should emit Transfer event
    totalSupply
      ✓ should return the total amount of tokens
    DOMAIN_SEPARATOR
      ✓ should be keccak256 of EIP712 domain struct
    balanceOf
      ✓ should return the total amount of tokens
    transfer
      ✓ should transfer the requested amount
      ✓ should emit a transfer event
    transferFrom
      ✓ should transfer the requested amount
      ✓ should emit a transfer event
    approve
      ✓ should approve the requested amount
      ✓ should emit an approval event
    burn
      ✓ should decrement account's balance
```

```
      ✓ should emit Transfer event
    burnFrom
      ✓ should decrement account's balance
      ✓ should decrement allowance
      ✓ should emit Transfer event
    permit
      ✓ should emit an approval event
      ✓ should approve the requested amount

  L2WormholeGateway
    initialization
      ✓ should set the wormhole bridge address
      ✓ should set the wormhole bridge token address
      ✓ should set the canonical tBTC address
    receiveTbtc
      when receiver is the zero address
        ✓ should revert
      when the transferred amount is zero
        ✓ should revert
      when receiver is non-zero address
        when the minting limit was not reached
          ✓ should transfer wormhole tBTC to the contract
          ✓ should mint tBTC to the receiver
          ✓ should complete transfer with the bridge
          ✓ should emit the WormholeTbtcReceived event
          ✓ should increase the minted amount counter
        when the minting limit was reached
          ✓ should transfer wormhole tBTC to the contract
          ✓ should mint tBTC to the receiver before reaching the minting limit
          ✓ should send wormhole tBTC to the receiver after reaching the minting limit
          ✓ should increase the minted amount counter
    sendTbtc
      when there is not enough wormhole tBTC
        ✓ should revert
      when there is enough wormhole tBTC
        when the receiver address is zero
          ✓ should revert
        when the amount is zero
          ✓ should revert
        when the receiver address and amount are non-zero
          when the target chain has no tBTC gateway
            ✓ should burn canonical tBTC from the caller
            ✓ should approve burned amount of wormhole tBTC to the bridge
            ✓ should sent tokens through the bridge
            ✓ should emit the WormholeTbtcSent event
          when the target chain has a tBTC gateway
            ✓ should burn canonical tBTC from the caller
            ✓ should approve burned amount of wormhole tBTC to the bridge
            ✓ should sent tokens through the bridge
            ✓ should emit the WormholeTbtcSent event
          when the amount is below dust
            ✓ should revert
          when the amount is just above the dust
            ✓ should burn canonical tBTC from the caller
            ✓ should approve burned amount of wormhole tBTC to the bridge
            ✓ should sent the entire amount through the bridge
          when the amount has a small dust
            ✓ should burn canonical tBTC from the caller after dropping dust
            ✓ should approve burned amount of wormhole tBTC to the bridge after dropping dust
            ✓ should drop the dust before sending over the bridge
          when the amount has a lot of dust
            ✓ should burn canonical tBTC from the caller after dropping dust
            ✓ should approve burned amount of wormhole tBTC to the bridge after dropping dust
            ✓ should drop the dust before sending over the bridge
    updateGatewayAddress
      when called by a third party
        ✓ should revert
      when called by the governance
        ✓ should update the gateway address
        ✓ should emit the GatewayAddressUpdated event
      when disabling gateway
        ✓ should update the gateway address
        ✓ should emit the GatewayAddressUpdated event
    updateMintingLimit
      when called by a third party
        ✓ should revert
      when called by the governance
        ✓ should update the minting limit
        ✓ should emit the MintingLimitUpdated event
    toWormholeAddress
      ✓ should convert Ethereum address into Wormhole format
    fromWormholeAddress
      ✓ should convert Wormhole address into Ethereum format

  MaintainerProxy
    requestNewWallet
      when called by an unauthorized third party
        ✓ should revert
      when called by an SPV maintainer that is not wallet maintainer
        ✓ should revert
      when called by a wallet maintainer
        ✓ should emit NewWalletRequested event
        ✓ should refund ETH
    submitDepositSweepProof
      when called by an unauthorized third party
        ✓ should revert
      when called by a wallet maintainer that is not SPV maintainer
        ✓ should revert
      when called by an SPV maintainer
        when there is only one input
          when the single input is a revealed unswept P2SH deposit
            ✓ should emit DepositSwept event
            ✓ should refund ETH
          when the single input is a revealed unswept P2WSH deposit
            ✓ should emit DepositSwept event
            ✓ should refund ETH
          when the single input is a revealed unswept deposit with a trusted vault
            ✓ should emit DepositSwept event
            ✓ should refund ETH
          when the single input is a revealed unswept deposit with a non-trusted vault
            ✓ should emit DepositSwept event
            ✓ should refund ETH
        when there are multiple inputs
          when input vector consists only of revealed unswept deposits and the expected main UTXO
            ✓ should emit DepositSwept event
            ✓ should refund ETH
          when input vector consists only of revealed unswept deposits with a trusted vault and the
expected main UTXO
            ✓ should emit DepositSwept event
            ✓ should refund ETH
          when input vector consists only of revealed unswept deposits with a non-trusted vault and the
expected main UTXO
            ✓ should emit DepositSwept event
            ✓ should refund ETH
          when input vector consists only of revealed unswept deposits with different trusted vaults and
the expected main UTXO
            ✓ should emit DepositSwept event
            ✓ should refund ETH
          when input vector consists only of revealed unswept deposits but there is no main UTXO since it
is not expected
            ✓ should emit DepositSwept event
            ✓ should refund ETH
    submitRedemptionProof
      when called by an unauthorized third party
        ✓ should revert
      when called by a wallet maintainer that is not SPV maintainer
        ✓ should revert
      when called by an SPV maintainer
        when there is only one output
          when the single output is a pending requested redemption
            ✓ should emit RedemptionsCompleted event
            ✓ should refund ETH
          when the single output is a non-reported timed out requested redemption
            ✓ should emit RedemptionsCompleted event
            ✓ should refund ETH
          when the single output is a reported timed out requested redemption
            ✓ should emit RedemptionsCompleted event
            ✓ should refund ETH
        when there are multiple outputs
          when output vector consists only of pending requested redemptions
            ✓ should emit RedemptionsCompleted event
            ✓ should refund ETH
          when output vector consists of pending requested redemptions and a non-zero change
            ✓ should emit RedemptionsCompleted event
            ✓ should refund ETH
          when output vector consists only of reported timed out requested redemptions
            ✓ should emit RedemptionsCompleted event
            ✓ should refund ETH
          when output vector consists of reported timed out requested redemptions and a non-zero change
            ✓ should emit RedemptionsCompleted event
            ✓ should refund ETH
          when output vector consists of pending requested redemptions and reported timed out requested
redemptions
            ✓ should emit RedemptionsCompleted event
            ✓ should refund ETH
          when output vector consists of pending requested redemptions, reported timed out requested
redemptions and a non-zero change
            ✓ should emit RedemptionsCompleted event
            ✓ should refund ETH
    notifyWalletCloseable
      when called by an unauthorized third party
        ✓ should revert
      when called by an SPV maintainer that is not wallet maintainer
        ✓ should revert
      when called by a wallet maintainer
        when wallet reached the maximum age
          when wallet balance is zero
            ✓ should emit WalletClosing event
            ✓ should refund ETH
          when wallet balance is greater than zero
            ✓ should emit WalletMovingFunds event
            ✓ should refund ETH
        when wallet did not reach the maximum age but their balance is lesser than the minimum threshold
          when wallet balance is zero
            ✓ should emit WalletClosing event
            ✓ should refund ETH
          when wallet balance is greater than zero
            ✓ should emit WalletMovingFunds event
            ✓ should refund ETH
    defeatFraudChallenge
      when the input is non-witness
        when the transaction has single input
          when the input is marked as correctly spent in the Bridge
            ✓ should emit FraudChallengeDefeated event
            ✓ should refund ETH
        when the transaction has multiple inputs
          when the input is marked as correctly spent in the Bridge
            ✓ should emit FraudChallengeDefeated event
            ✓ should refund ETH
      when the input is witness
        when the transaction has single input
          when the input is marked as correctly spent in the Bridge
            ✓ should emit FraudChallengeDefeated event
            ✓ should refund ETH
        when the transaction has multiple inputs
          when the input is marked as correctly spent in the Bridge
            ✓ should emit FraudChallengeDefeated event
            ✓ should refund ETH
    defeatFraudChallengeWithHeartbeat
      ✓ should emit FraudChallengeDefeated event
      ✓ should refund ETH
    submitMovingFundsProof
      when called by an unauthorized third party
        ✓ should revert
      when called by a wallet maintainer that is not SPV maintainer
        ✓ should revert
      when called by an SPV maintainer
        when there is a single target wallet
          ✓ should emit MovingFundsCompleted event
          ✓ should refund ETH
        when there are multiple target wallets and the amount is indivisible
          ✓ should emit MovingFundsCompleted event
          ✓ should refund ETH
        when there are multiple target wallets and the amount is divisible
          ✓ should emit MovingFundsCompleted event
          ✓ should refund ETH
    resetMovingFundsTimeout
      ✓ should emit MovingFundsTimeoutReset event
      ✓ should refund ETH
    notifyMovingFundsBelowDust
      when called by an unauthorized third party
        ✓ should revert
      when called by an SPV mantainer that is not wallet maintainer
        ✓ should revert
      when called by a wallet maintainer
        ✓ should emit MovingFundsBelowDustReported event
        ✓ should refund ETH
    submitMovedFundsSweepProof
      when called by an unauthorized third party
        ✓ should revert
      when called by a wallet maintainer that is not SPV maintainer
        ✓ should revert
      when called by an SPV maintainer
        when the sweeping wallet has no main UTXO set
          when there is a single input referring to a Pending sweep request
            ✓ should emit MovedFundsSwept event
            ✓ should refund ETH
        when the sweeping wallet has a main UTXO set
          when the first input refers to a Pending sweep request and the second input refers to the
sweeping wallet main UTXO
            ✓ should emit MovedFundsSwept event
            ✓ should refund ETH
    notifyWalletClosingPeriodElapsed
      when called by an unauthorized third party
        ✓ should revert
      when called by an SPV maintainer that is not wallet maintainer
        ✓ should revert
      when called by a wallet maintainer
        ✓ should emit WalletClosed event
        ✓ should refund ETH
    authorizeWalletMaintainer
      when the caller is not the owner
        ✓ should revert
      when the caller is the owner
        ✓ should be already populated with the authorized maintainer
        ✓ should authorize a thirdParty
        ✓ should be total of 2 authorized maintainers
        ✓ should add a thirdParty to a maintainers list
        ✓ should emit a WalletMaintainerAuthorized event
    authorizeSpvMaintainer
      when the caller is not the owner
        ✓ should revert
      when the caller is the owner
        ✓ should be already populated with the authorized maintainer
        ✓ should authorize a thirdParty
        ✓ should be total of 2 authorized maintainers
        ✓ should add a thirdParty to a maintainers list
        ✓ should emit an SpvMaintainerAuthorized event
    unauthorizeWalletMaintainer
      when the caller is not the owner
        ✓ should revert
      when the caller is the owner
        ✓ should be a total of 0 authorized maintainers
        when there are no authorized maintainers
          ✓ should revert
        when there are authorized maintainers
          when maintainer to unauthorize is not among the authorized maintainers
            ✓ should revert
        when there is one authorized maintainer
          when unauthorizing the one that is authorized
            ✓ should unauthorize the maintainer
            ✓ should emit a WalletMaintainerUnauthorized event
        when there are many authorized maintainers
          when unauthorizing a couple of maintainers from the beginning
            ✓ should unauthorize the maintainer
            ✓ should change the last maintainer's index with the unauthorized one
            ✓ should unauthorize the other maintainer
            ✓ should change the last maintainer's index with the unauthorized one
            ✓ should remove 2 maintainers from the maintainers array
            ✓ should emit a WalletMaintainerUnauthorized event
            ✓ should emit a WalletMaintainerUnauthorized event
          when unauthorizing a couple of maintainers from the middle
            ✓ should unauthorize a maintainer
            ✓ should change the last maintainer's index with the unauthorized one
            ✓ should unauthorize the other maintainer
            ✓ should change the last maintainer's index with the unauthorized one
            ✓ should remove 2 maintainers from the maintainers array
            ✓ should emit a WalletMaintainerUnauthorized event
            ✓ should emit a WalletMaintainerUnauthorized event
          when unauthorizing a couple of maintainers from the end
            ✓ should unauthorize a maintainer
            ✓ should unauthorize the other maintainer
            ✓ should change the last maintainer's index with the unauthorized one
            ✓ should remove 2 maintainers from the maintainers array
            ✓ should emit a WalletMaintainerUnauthorized event
            ✓ should emit a WalletMaintainerUnauthorized event
    unauthorizeSpvMaintainer
      when the caller is not the owner
        ✓ should revert
      when the caller is the owner
        ✓ should be a total of 0 authorized maintainers
        when there are no authorized maintainers
          ✓ should revert
        when there are authorized maintainers
          when maintainer to unauthorize is not among the authorized maintainers
            ✓ should revert
        when there is one authorized maintainer
          when unauthorizing the one that is authorized
            ✓ should unauthorize the maintainer
            ✓ should emit an SpvMaintainerUnauthorized event
        when there are many authorized maintainers
          when unauthorizing a couple of maintainers from the beginning
            ✓ should unauthorize the maintainer
            ✓ should change the last maintainer's index with the unauthorized one
            ✓ should unauthorize the other maintainer
            ✓ should change the last maintainer's index with the unauthorized one
            ✓ should remove 2 maintainers from the maintainers array
            ✓ should emit an SpvMaintainerUnauthorized event
            ✓ should emit an SpvMaintainerUnauthorized event
          when unauthorizing a couple of maintainers from the middle
            ✓ should unauthorize a maintainer
            ✓ should change the last maintainer's index with the unauthorized one
            ✓ should unauthorize the other maintainer
            ✓ should change the last maintainer's index with the unauthorized one
            ✓ should remove 2 maintainers from the maintainers array
            ✓ should emit an SpvMaintainerUnauthorized event
            ✓ should emit an SpvMaintainerUnauthorized event
          when unauthorizing a couple of maintainers from the end
            ✓ should unauthorize a maintainer
            ✓ should unauthorize the other maintainer
            ✓ should change the last maintainer's index with the unauthorized one
            ✓ should remove 2 maintainers from the maintainers array
            ✓ should emit an SpvMaintainerUnauthorized event
            ✓ should emit an SpvMaintainerUnauthorized event
    updateBridge
      when called by a third party
        ✓ should revert
      when called by the owner
        ✓ should update the bridge
        ✓ should emit the BridgeUpdated event
    updateGasOffsetParameters
      when called by a third party
        ✓ should revert
      when called by the owner
        ✓ should emit the GasOffsetParametersUpdated event
        ✓ should update submitRedemptionProofGasOffset
        ✓ should update resetMovingFundsTimeoutGasOffset
        ✓ should update submitMovingFundsProofGasOffset
        ✓ should update notifyMovingFundsBelowDustGasOffset
        ✓ should update submitMovedFundsSweepProofGasOffset
        ✓ should update requestNewWalletGasOffset
        ✓ should update notifyWalletCloseableGasOffset
        ✓ should update notifyWalletClosingPeriodElapsedGasOffset
        ✓ should update defeatFraudChallengeGasOffset
        ✓ should update defeatFraudChallengeWithHeartbeatGasOffset
    updateReimbursementPool
      when called by a third party
        ✓ should revert
      when called by the owner
        ✓ should emit the ReimbursementPoolUpdated event

LightRelay
  genesis
    when called with valid inputs
      ✓ should record the relay as ready for use
      ✓ should emit the Genesis event
      ✓ should record the genesis epoch difficulty correctly
    when called with invalid block height
      ✓ should revert
    when called with invalid header data
      ✓ should revert
    when called with excessive proof length
      ✓ should revert
    when called with zero proof length
      ✓ should revert
    when called by anyone other than governance
      ✓ should revert
    when called more than once
      ✓ should revert
  setProofLength
    before genesis
      ✓ should revert
    after genesis
      when called correctly
        ✓ should store the new proof length
        ✓ should emit the ProofLengthChanged event
      when called with excessive proof length
        ✓ should revert
      when called with zero proof length
        ✓ should revert
      when called with unchanged proof length
        ✓ should revert
      when called by anyone other than governance
        ✓ should revert
  authorizations
    authorization status
      ✓ should start at false
      when set by governance
        ✓ should be updated
        ✓ should emit an event
      when set by someone other than governance
        ✓ should revert
      when unset by governance
        ✓ should be updated
        ✓ should emit an event
    submitter authorization
      ✓ should start at false
      when set by governance
        ✓ should be updated
        ✓ should emit an event
      when set by someone other than governance
        ✓ should revert
        when unset by governance
          ✓ should be updated
          ✓ should emit an event
    retarget
      when called before genesis
        ✓ should revert
      after genesis (epoch 274)
        when called correctly
          ✓ should store the new difficulty
          ✓ should emit the Retarget event
        with incorrect number of headers
          ✓ should revert
        with too few headers before retarget
          ✓ should revert
        with too few headers after retarget
          ✓ should revert
        with proof length 9
          ✓ should store the new difficulty
          ✓ should emit the Retarget event
        with appropriate authorisation
          ✓ should store the new difficulty
          ✓ should emit the Retarget event
        without appropriate authorisation
          ✓ should revert
      after genesis (invalid)
        ✓ should reject chains with invalid difficulty
      after genesis (long chain)
        with proof length 6
          ✓ should store the new difficulty
          ✓ should emit the Retarget event
        with proof length 50
          ✓ should store the new difficulty
          ✓ should emit the Retarget event
    validateChain
      when called before genesis
        ✓ should revert
      when called after genesis (epoch 274)
        ✓ should accept valid header chains
        ✓ should accept short header chains
        ✓ should accept long header chains
        ✓ should reject single headers
        ✓ should reject header chains with an unknown retarget
        ✓ should reject header chains in a future epoch
      when called after genesis (epoch 275)
        ✓ should accept valid header chains
        ✓ should reject header chains partially in a past epoch
        ✓ should reject header chains fully in a past epoch
      when called after a retarget
        in the genesis epoch
          ✓ should accept valid header chains
        over the retarget
          ✓ should accept valid header chains (3 before, 1 after)
          ✓ should accept valid header chains (2 before, 2 after)
          ✓ should accept valid header chains (1 before, 3 after)
        in the new epoch
          ✓ should accept valid header chains
      with chain reorgs
        valid chains
          ✓ should be accepted
        invalid chains
          ✓ should be rejected
      gas costs
        with proof length 6
          ✓ should accept valid header chains
        with proof length 18
          ✓ should accept valid header chains
    getBlockDifficulty
      when called before genesis
        ✓ should revert
      when called after genesis
        ✓ should return the difficulty for the first block of the epoch
        ✓ should return the difficulty for the last block of the epoch
```

```
        ✓ should revert for blocks before genesis
        ✓ should revert for blocks after the latest epoch
      when called after a retarget
        ✓ should return the difficulty for the first block of the genesis epoch
        ✓ should return the difficulty for the last block of the genesis epoch
        ✓ should return the difficulty for the first block of the next epoch
        ✓ should return the difficulty for the last block of the next epoch
        ✓ should revert for blocks before genesis
        ✓ should revert for blocks after the latest epoch
    getEpochDifficulty
      when called before genesis
        ✓ should revert
      when called after genesis
        ✓ should return the difficulty for the genesis epoch
        ✓ should revert for epochs before genesis
        ✓ should revert for unproven epochs
      when called after a retarget
        ✓ should return the difficulty for the genesis epoch
        ✓ should return the difficulty for the next epoch
        ✓ should revert for epochs before genesis
        ✓ should revert for unproven epochs
    getRelayRange
      when called before genesis
        ✓ should return nonsense
      when called after genesis
        ✓ should return a single epoch
      when called after a retarget
        ✓ should return two epochs
    getCurrentEpochDifficulty
      when called before genesis
        ✓ should return zero
      when called after genesis
        ✓ should return the difficulty for the genesis epoch
      when called after a retarget
        ✓ should return the difficulty for the next epoch
    getPrevEpochDifficulty
      when called before genesis
        ✓ should return zero
      when called after genesis
        ✓ should return zero
      when called after a retarget
        ✓ should return the difficulty for the genesis epoch
    getCurrentAndPrevEpochDifficulty
      when called before genesis
        ✓ should return zero for both
      when called after genesis
        ✓ should return current difficulty, and zero for previous
      when called after a retarget
        ✓ should return current and previous difficulty

  LightRelayMaintainerProxy
    authorize
      when called by non-owner
        ✓ should revert
      when called by the owner
        when the maintainer is already authorized
          ✓ should revert
        when the maintainer is not authorized yet
          ✓ should authorize the address
          ✓ should emit the MaintainerAuthorized event
    deauthorize
      when called by non-owner
        ✓ should revert
      when called by the owner
        when the maintainer is not authorized
          ✓ should revert
        when the maintainer is authorized
          ✓ should deauthorize the address
          ✓ should emit the MaintainerDeauthorized event
    updateLightRelay
      when called by non-owner
        ✓ should revert
      when called by the owner
        when called with zero address
          ✓ should revert
        when called with a non-zero address
          ✓ should update the light relay address
          ✓ should emit the LightRelayUpdated event
    updateReimbursementPool
      when called by non-owner
        ✓ should revert
      when called by the owner
        ✓ should emit the ReimbursementPoolUpdated event
    updateRetargetGasOffset
      when called by non-owner
        ✓ should revert
      when called by the owner
        ✓ should emit the RetargetGasOffsetUpdated event
        ✓ should update retargetGasOffset
    retarget
      when called by an unauthorized address
        ✓ should revert
      when called by an authorized maintainer
        when the proof length is 10 headers
          ✓ should emit Retarget event
          ✓ should refund ETH
        when the proof length is 20 headers
          ✓ should emit Retarget event
          ✓ should refund ETH
        when the proof length is 50 headers
          ✓ should emit Retarget event
          ✓ should refund ETH

  DonationVault
    constructor
      when called with a 0-address bank
        ✓ should revert
      when called with correct parameters
        ✓ should set the Bank field
    donate
      when caller has not enough balance in the bank
        ✓ should revert
      when vault does not have enough allowance for caller's balance
        ✓ should revert
      when called with correct parameters
        ✓ should decrease donor's balance
        ✓ should not increase vault's balance
        ✓ should emit BalanceDecreased event
        ✓ should emit DonationReceived event
    receiveBalanceApproval
      when called not by the bank
        ✓ should revert
      when caller has not enough balance in the bank
        ✓ should revert
      when called with correct parameters
        ✓ should decrease donor's balance
        ✓ should not increase vault's balance
        ✓ should emit BalanceDecreased event
        ✓ should emit DonationReceived event
    receiveBalanceIncrease
      when called not by the bank
        ✓ should revert
      when called with no depositors
        ✓ should revert
      when called with correct parameters
        ✓ should not increase depositors' balances
        ✓ should not increase vault's balance
        ✓ should emit BalanceDecreased event
        ✓ should emit DonationReceived event

  TBTCVault - OptimisticMinting
    requestOptimisticMint
      when called not by a minter
        ✓ should revert
      when called by a minter
        when optimistic minting is paused
          ✓ should revert
        when optimistic minting has been already requested
          ✓ should revert
        when the deposit has not been revealed
          ✓ should revert
        when the deposit has been revealed
          when the deposit has been swept
            ✓ should revert
          when the deposit is targeted to another vault
            ✓ should revert
          when all conditions are met
            ✓ should request optimistic minting
            ✓ should emit an event
    finalizeOptimisticMint
      when called not by a minter
        ✓ should revert
      when called by a minter
        when optimistic minting is paused
          ✓ should revert
        when minting has not been requested
          ✓ should revert
        when the minting delay has not passed yet
          ✓ should revert
        when requested minting has been already finalized
          ✓ should revert
        when the deposit has been already swept
          ✓ should revert
        when all conditions are met
          when fees are non-zero
            ✓ should send optimistic mint fee to treasury
            ✓ should mint TBTC to depositor
            ✓ should incur optimistic mint debt
            ✓ should mark the request as finalized
            ✓ should emit an event
          when the optimistic minting fee is zero
            ✓ should send no optimistic mint fee to treasury
            ✓ should mint TBTC to depositor
            ✓ should incur optimistic mint debt
            ✓ should mark the request as finalized
            ✓ should emit an event
          when the bridge deposit treasury fee is zero
            ✓ should send optimistic mint fee to treasury
            ✓ should mint TBTC to depositor
            ✓ should incur optimistic mint debt
            ✓ should mark the request as finalized
            ✓ should emit an event
          when both fees are zero
            ✓ should mint TBTC to depositor
            ✓ should incur optimistic mint debt
            ✓ should mark the request as finalized
            ✓ should emit an event
    cancelOptimisticMint
      when called not by a guardian
        ✓ should revert
      when called by a guardian
        when minting has not been requested
          ✓ should revert
        when requested minting has been finalized
          ✓ should revert
        when requested minting has not been finalized
          ✓ should cancel optimistic minting
          ✓ should emit an event
    addMinter
      when called not by the governance
        ✓ should revert
      when called by the governance
        when address is not a minter
          ✓ should add address as a minter
          ✓ should emit an event
        when address is a minter
```

```
        ✓ should revert
      when there are multiple minters
        ✓ should add them into the list
  removeMinter
    when called not by the governance or a guardian
      ✓ should revert
    when called by the governance
      when address is a minter
        ✓ should take minter role from the address
        ✓ should emit an event
      when address is not a minter
        ✓ should revert
    when called by a guardian
      when address is not a minter
        ✓ should revert
      when address is a minter
        ✓ should take minter role from the address
        ✓ should emit an event
      when there are multiple minters
        when deleting the first minter
          ✓ should update the minters list
        when deleting the last minter
          ✓ should update the minters list
        when deleting minter from the middle of the list
          ✓ should update the minters list
  addGuardian
    when called not by the governance
      ✓ should revert
    when called by the governance
      when address is not a guardian
        ✓ should add address as a guardian
        ✓ should emit an event
      when address is a guardian
        ✓ should revert
  removeGuardian
    when called not by the governance
      ✓ should revert
    when called by the governance
      when address is a guardian
        ✓ should take guardian role from the address
        ✓ should emit an event
      when address is not a guardian
        ✓ should revert
  pauseOptimisticMinting
    when called not by the governance
      ✓ should revert
    when called by the governance
      when optimistic minting is already paused
        ✓ should revert
      when optimistic minting is not paused
        ✓ should pause optimistic minting
        ✓ should emit an event
  unpauseOptimisticMinting
    when called not by the governance
      ✓ should revert
    when called by the governance
      when optimistic minting is not paused
        ✓ should revert
      when optimistic minting is paused
        ✓ should unpause optimistic minting
        ✓ should emit an event
  beginOptimisticMintingFeeUpdate
    when called not by the governance
      ✓ should revert
    when called by the governance
      ✓ should not update the optimistic minting fee
      ✓ should start the governance delay timer
      ✓ should emit an event
  finalizeOptimisticMintingFeeUpdate
    when called not by the governance
      ✓ should revert
    when the update process is not initiated
```

```
        ✓ should revert
      when the governance delay has not passed
        ✓ should revert
      when the update process is initiated and governance delay passed
        ✓ should update the optimistic minting fee
        ✓ should emit an event
        ✓ should reset the governance delay timer
    beginOptimisticMintingDelayUpdate
      when called not by the governance
        ✓ should revert
      when called by the governance
        ✓ should not update the optimistic minting delay
        ✓ should start the governance delay timer
        ✓ should emit an event
    finalizeOptimisticMintingDelayUpdate
      when called not by the governance
        ✓ should revert
      when the update process is not initiated
        ✓ should revert
      when the governance delay has not passed
        ✓ should revert
      when the update process is initiated and governance delay passed
        ✓ should update the optimistic minting delay
        ✓ should emit an event
        ✓ should reset the governance delay timer
    calculateDepositKey
      ✓ should calculate the key as expected
      ✓ should calculate the same key as the Bridge
    receiveBalanceIncrease
      when the deposit for which optimistic minting was requested gets swept after finalization
        ✓ should repay optimistic minting debt
        ✓ should emit an event
      when multiple deposits gets swept after finalization
        when both deposits were optimistically minted
          ✓ should repay optimistic minting debt
          ✓ should mint the right amount of TBTC to depositor
          ✓ should emit an event
        when only one deposit was optimistically minted
          ✓ should repay optimistic minting debt
          ✓ should mint the right amount of TBTC
          ✓ should emit an event

  TBTCVault - Redemption
    unmintAndRedeem
      when the redeemer has no TBTC
        ✓ should revert
      when the redeemer has not enough TBTC
        ✓ should revert
      when there is a single redeemer
        ✓ should transfer balances to Bridge
        ✓ should request redemptions in Bridge
        ✓ should burn TBTC
        ✓ should emit Unminted events
      when amount is not fully convertible to satoshis
        ✓ should transfer balances to Bridge
        ✓ should request redemptions in Bridge
        ✓ should burn TBTC
        ✓ should emit Unminted events
      when there are multiple redeemers
BigNumber { value: "1000000000000000000000" }
BigNumber { value: "1000000000000000000000" }
        ✓ should transfer balances to Bridge
        ✓ should request redemptions in Bridge
        ✓ should burn TBTC
        ✓ should emit Unminted events
    receiveApproval
      when called via approveAndCall
        when called with non-empty extraData
          when there is a single redeemer
            ✓ should transfer balances to Bridge
            ✓ should request redemptions in Bridge
            ✓ should burn TBTC
          ✓ should emit Unminted events
        when there are multiple redeemers
          ✓ should transfer balances to Bridge
          ✓ should request redemptions in Bridge
          ✓ should burn TBTC
          ✓ should emit Unminted events

  TBTCVault
    constructor
      when called with a 0-address bank
        ✓ should revert
      when called with a 0-address TBTC token
        ✓ should revert
      when called with a 0-address bridge
        ✓ should revert
      when called with correct parameters
        ✓ should set the Bank field
        ✓ should set the TBTC token field
    recoverERC20FromToken
      when called not by the governance
        ✓ should revert
      when called with correct parameters
        ✓ should do a successful recovery
    recoverERC721FromToken
      when called not by the governance
        ✓ should revert
      when called with correct parameters
        ✓ should do a successful recovery
    recoverERC20
      when called not by the governance
        ✓ should revert
      when called with correct parameters
        ✓ should do a successful recovery
    recoverERC721
      when called not by the governance
        ✓ should revert
      when called with correct parameters
        ✓ should do a successful recovery
    mint
      when minter has not enough balance in the bank
        ✓ should revert
      when there is a single minter
        ✓ should transfer balance to the vault
        ✓ should mint TBTC
        ✓ should emit Minted event
      when amount is not fully convertible to satoshis
        ✓ should transfer balance to the vault
        ✓ should mint TBTC
        ✓ should emit Minted event
      when there are multiple minters
        ✓ should transfer balances to the vault
        ✓ should mint TBTC
        ✓ should emit Minted event
    unmint
      when the unminter has no TBTC
        ✓ should revert
      when the unminter has not enough TBTC
        ✓ should revert
      when there is a single unminter
        ✓ should transfer balance to the unminter
        ✓ should burn TBTC
        ✓ should emit Unminted events
      when amount is not fully convertible to satoshis
        ✓ should transfer balance to the unminter
        ✓ should burn TBTC
        ✓ should emit Unminted events
      when there are multiple unminters
        ✓ should transfer balances to unminters
        ✓ should burn TBTC
        ✓ should emit Unminted events
    receiveApproval
      when called not for TBTC token
        ✓ should revert
      when called directly
        ✓ should revert
      when called via approveAndCall
        when called with an empty extraData
          ✓ should transfer balance to the unminter
          ✓ should burn TBTC
          ✓ should emit Unminted event
        when amount is not fully convertible to satoshis
          ✓ should transfer balance to the unminter
          ✓ should burn TBTC
          ✓ should emit Unminted events
    receiveBalanceApproval
      when called not by the bank
        ✓ should revert
      when caller has not enough balance in the bank
        ✓ should revert
      when there is a single caller
        ✓ should transfer balance to the vault
        ✓ should mint TBTC
        ✓ should emit Minted event
      when there are multiple callers
        ✓ should transfer balances to the vault
        ✓ should mint TBTC
        ✓ should emit Minted event
    receiveBalanceIncrease
      when called not by the bank
        ✓ should revert
      when called with no depositors
        ✓ should revert
      with single depositor
        ✓ should mint TBTC
        ✓ should emit Minted event
      with multiple depositors
        ✓ should mint TBTC
        ✓ should emit Minted events
    initiateUpgrade
      when called not by the governance
        ✓ should revert
      when called by the governance
        when called with a zero-address new vault
          ✓ should revert
        when called with a non-zero-address new vault
          ✓ should not transfer TBTC token ownership
          ✓ should set the upgrade initiation time
          ✓ should set the new vault address
          ✓ should emit UpgradeInitiated event
    finalizeUpgrade
      when called not by the governance
        ✓ should revert
      when called by the governance
        when the upgrade process has not been initiated
          ✓ should revert
        when the upgrade process has been initiated
          when the governance delay has not passed
            ✓ should revert
          when the governance delay passed
            ✓ should transfer TBTC token ownership
            ✓ should transfer the entire bank balance
            ✓ should emit UpgradeFinalized event
            ✓ should reset the upgrade initiation time
            ✓ should reset the new vault address
    amountToSatoshis
      when the amount is convertible with a remainder
        ✓ should calculate correct convertible amount
        ✓ should calculate correct remainder
        ✓ should calculate correct satoshi amount
      when the amount is convertible without a remainder
        ✓ should calculate correct convertible amount
        ✓ should calculate correct remainder
        ✓ should calculate correct satoshi amount
    2190 passing (3m)
    27 pending

Integration tests for 'keep-network/tbtc-v2'

  Integration Test - Full flow
transferred 4500000000 T to the VendingMachine for KEEP
transferred 4500000000 T to the VendingMachine for NU
Warning: Potentially unsafe deployment of WalletRegistry

    You are using the `unsafeAllow.external-library-linking` flag to include external libraries.
    Make sure you have manually checked that the linked libraries are upgrade safe.

Warning: Potentially unsafe deployment of BridgeStub

    You are using the `unsafeAllow.external-library-linking` flag to include external libraries.
    Make sure you have manually checked that the linked libraries are upgrade safe.

Initialized Wallet Owner address: 0x3c705dB336C81c7FEFC5746e283aB2c0781A4B7b in transaction:
0x4c54557085513b45258fe2a2f2b11d7b8abe6f870942f0d513209c4d26df7624
    Check deposit and redemption flow
      when wallet is created
        when a deposit is revealed
          ✓ should create a deposit
        when the deposit sweep proof is submitted
          ✓ should mint TBTC tokens for the depositor
          ✓ should increase the balance of vault in the bank
          ✓ should update the main UTXO of the wallet
        when a redemption is requested
          ✓ should create a pending redemption request
          ✓ should increase the pending redemptions value of the wallet
          ✓ should increase the balance of bridge in the bank
        when the redemption proof is submitted
          ✓ should zero the pending redemptions value of the wallet
          ✓ should zero the balance of bridge in the bank
          ✓ should update the main UTXO of the wallet

  Integration Test - Slashing
    notifyFraudChallengeDefeatTimeout
      when wallet is created
        when a fraud is reported
          ✓ should slash wallet members
          ✓ should close the wallet in the wallet registry
          ✓ should terminate the wallet in the bridge
          ✓ should consume around 3 100 000 gas for Bridge.notifyMovingFundsTimeoutTx transaction
    notifyRedemptionTimeout
      when wallet is created
        when a redemption timeout is reported
          ✓ should slash wallet members
          ✓ should not close the wallet in the wallet registry
          ✓ should transition the wallet in the bridge to the MovingFunds state
          ✓ should consume around 3 150 000 gas for Bridge.notifyRedemptionTimeout transaction
    notifyMovingFundsTimeout
      when wallet is created
        when moving funds timeout is reported
          ✓ should slash wallet members
          ✓ should close the wallet in the wallet registry
          ✓ should terminate the wallet in the bridge
          ✓ should consume around 3 100 000 gas for Bridge.notifyMovingFundsTimeoutTx transaction

  Integration Test - Wallet Creation
    new wallet creation (happy path)
      ✓ should register a new wallet in the WalletRegistry
      ✓ should register a new wallet details in the Bridge
      ✓ should register a new wallet as active in the Bridge
      ✓ should consume around 94 000 gas for Bridge.requestNewWallet transaction
      ✓ should consume around 341 000 gas for WalletRegistry.approveDkgResult transaction

  27 passing (1m)

Tests for 'thesis/mezo-portal'
```

```
BitcoinDepositor
  initialize
    when called directly on the implementation
      ✔ should revert
    when called on the proxy
      when called again
        ✔ should revert (78ms)
      when called with zero-address bridge
        ✔ should revert
      when called with zero-address tBTC vault
        ✔ should revert
      when called with zero-address tBTC token
        ✔ should revert
      when called with zero-address portal
        ✔ should revert
  initializeDeposit
    when the deposit owner is zero address
      ✔ should revert
    when the deposit was already initialized
      ✔ should revert
    when initializing for the first time
      ✔ should set the deposit state to Initialized
      ✔ should emit DepositInitialized event
  finalizeDeposit
    when the deposit was not initialized before
      ✔ should revert
    when the deposit was not finalized by the bridge
      ✔ should revert
    when deposit was finalized by the bridge
      when a single, non-lockable deposit was finalized
        when deposit owner param is different than during the initialization
          ✔ should revert
        when deposit lock period param is different than during the initialization
          ✔ should revert
        when called with the same params as during the initialization
          ✔ should emit DepositFinalized event
          ✔ should set the deposit state to Finalized
          ✔ should deposit tokens to the Portal contract (59ms)
          ✔ should keep the surplus in the BitcoinDepositor contract
      when a single, lockable deposit was finalized
        when deposit owner param is different than during the initialization
          ✔ should revert
        when deposit lock period param is different than during the initialization
          ✔ should revert
        when called with the same params as during the initialization
          ✔ should emit DepositFinalized event
          ✔ should set the deposit state to Finalized
          ✔ should deposit tokens to the Portal contract
          ✔ should keep the surplus in the BitcoinDepositor contract
      when multiple deposits were finalized
        when called with the same params as during the initialization
          ✔ should set the states of deposits to Finalized
          ✔ should deposit tokens to the Portal contract
          ✔ should keep the surplus in the BitcoinDepositor contract
      when called for the same deposit second time
        when called with the same params
          ✔ should revert
        when called with different params
          ✔ should revert

Portal - deposit method
  deposit
    when called incorrectly
      when depositing without locking
        when depositing 0-address token
          ✔ should revert
        when depositing unsupported token
          ✔ should revert
        when depositing 0 amount
          ✔ should revert
      when depositing with locking
        when token is not supported
```

```
          ✔ should revert
        when token is not lockable
          ✔ should revert
        when lock time is less than 1 week
          ✔ should revert
        when lock time is less than min lock time
          ✔ should revert
        when lock time is greater than max lock time
          ✔ should revert
        when lock time is not a multiple of a week
          ✔ should round the lock period to the nearest week
    when called correctly
      when depositing without locking
        when depositing already supported token
          ✔ should emit a Deposited event
          ✔ should update the balance of the depositor
          ✔ should set unlock time to current block
          ✔ should transfer the token to the contract
        when depositing newly added supported token
          ✔ should emit a Deposited event
          ✔ should update the balance of the depositor
          ✔ should set unlock time to current block
          ✔ should transfer the token to the contract
      when depositing with locking
        ✔ should emit a Deposited event
        ✔ should emit a Locked event
        ✔ should set unlock time correctly

Portal - depositFor method
  depositFor
    when called incorrectly
      when depositing without locking
        when depositing 0-address token
          ✔ should revert
        when depositing unsupported token
          ✔ should revert
        when depositing with 0-address deposit owner
          ✔ should revert
        when depositing 0 amount
          ✔ should revert
      when depositing with locking
        when token is not supported
          ✔ should revert
        when token is not lockable
          ✔ should revert
        when lock time is less than 1 week
          ✔ should revert
        when lock time is less than min lock time
          ✔ should revert
        when lock time is greater than max lock time
          ✔ should revert
        when lock time is not a multiple of a week
          ✔ should round the lock period to the nearest week
    when called correctly
      when depositing without locking
        when depositing for someone else
          ✔ should emit a Deposited event
          ✔ should update the balance of the depositor
          ✔ should set unlock time to current block
          ✔ should transfer the token to the contract
        when depositing for oneself
          ✔ should emit a Deposited event
          ✔ should update the balance of the depositor
          ✔ should set unlock time to current block
          ✔ should transfer the token to the contract
      when depositing with locking
        when depositing with locking for someone else
          ✔ should emit a Deposited event
          ✔ should emit a Locked event
          ✔ should set unlock time correctly
        when depositing with locking for oneself
          ✔ should emit a Deposited event
          ✔ should emit a Locked event
          ✔ should set unlock time correctly

Portal - lock method
  lock
    when called incorrectly
      when the deposit doesn't exist
        ✔ should revert
      when token can't be locked
        ✔ should revert
      when lock period is less than 1 week
        ✔ should revert
      when lock period is less than min lock period
        ✔ should revert
      when lock period is more than max lock period
        ✔ should revert
      when lock period is less than current lock period
        ✔ should revert
    when called correctly
      when locking deposit for the first time
        ✔ should emit a Locked event
        ✔ should set unlock time correctly
      when extending lock period
        ✔ should emit a Locked event
        ✔ should set unlock time correctly
      when locking deposit after lock period has expired
        ✔ should emit a Locked event
        ✔ should set unlock time correctly

Portal - receiveApproval method
  receiveApproval
    when called incorrectly
      when depositing without locking
        when receiving unsupported token
          ✔ should revert
        when called directly
          ✔ should revert
        when receiving empty lock period data
          ✔ should revert
        when depositing amount exceeding uint96
          ✔ should revert
      when depositing with locking
        when it's trying to lock not lockable token
          ✔ should revert
        when it's trying to lock without lock period
          ✔ should revert
        when it's trying to lock with lock period less than min lock period
          ✔ should revert
        when it's trying to lock with lock period exceeding max lock period
          ✔ should revert
        when it's trying to lock with lock period of 1 day
          ✔ should revert
    when called correctly
      when depositing without locking
        ✔ should emit a Deposited event
        ✔ should update the balance of the depositor
        ✔ should transfer the token to the contract
      when depositing with locking
        ✔ should emit a Deposited event
        ✔ should emit a Locked event
        ✔ should set unlock time correctly

Portal - deployment and governance
  deployment
    when deployed with 0-address token
      ✔ should revert
    when deployed with supported tokens
      ✔ should deploy with 1 supported token
      ✔ should deploy with >1 supported tokens
  addSupportedToken
    when called by non-owner
      ✔ should revert
    when called by owner incorrectly
      when adding 0-address token
        ✔ should revert
      when adding already supported token
        ✔ should revert
    when called by owner correctly
      ✔ should emit a SupportedTokenAdded event
      ✔ should update the supported tokens
  setMinLockPeriod
    when called by non-owner
      ✔ should revert
    when called by owner
      when called incorrectly
        when trying to set min lock period greater than max lock period
          ✔ should revert
        when trying to set min lock period not normalized
          ✔ should revert
        when trying to set min lock period to value giving 0 post-normalization
          ✔ should revert
        when trying to set min lock period to 0
          ✔ should revert
      when called correctly
        ✔ should emit a MinLockPeriodUpdated event
        ✔ should update the min lock period
  setMaxLockPeriod
    when called by non-owner
      ✔ should revert
    when called by owner
      when called incorrectly
        when trying to set max lock period less than min lock period
          ✔ should revert
        when trying to set max lock period not normalized
          ✔ should revert
      when called correctly
        ✔ should emit a MaxLockPeriodUpdated event
        ✔ should update the max lock period

Portal - contract upgrades
  whan upgrading to an invalid contract
    when a variable was added before old variables
      ✔ should throw an error
    when a variable was removed
      ✔ should throw an error
  when upgrading to a valid contract
    ✔ new instance should have the same address as the old one
    contract variables
      ✔ should initialize new variable
      ✔ should add new supported tokens
      ✔ should keep old supported tokens
      ✔ should keep old supported tokens' balances
    contract functions
      ✔ should execute the new code
      ✔ should have access to storage slots from the previous version
      ✔ should be able to update storage slots

Portal - withdraw method
  withdraw
    when withdrawing incorrectly
      when token is not supported
        ✔ should revert
      when amount deposited is 0
        ✔ should revert
      when amount to withdraw is 0
        ✔ should revert
      when amount is greater than deposited balance
        ✔ should revert
    when deposit is not locked
      when the token being withdrawn is not lockable
        ✔ should emit Withdrawn event
        ✔ should decrease the deposited balance
        ✔ should transfer the token to the user
        ✔ should allow to withdraw the remaining balance later
```

```
          when the token being withdrawn is lockable
            ✔ should emit Withdrawn event
            ✔ should decrease the deposited balance
            ✔ should transfer the token to the user
            ✔ should allow to withdraw the remaining balance later
        when deposit is locked
          when the token being withdrawn is lockable
            when lock time has not passed
              when trying to withdraw lockable token
                ✔ should revert
            when lock time has passed
              ✔ should emit Withdrawn event
              ✔ should decrease the deposited balance
              ✔ should transfer the token to the user
              ✔ should allow to withdraw the remaining balance later
        when withdrawing funds deposited by someone else
          when called by the deposit funder
            ✔ should revert
          when called by the deposit owner
            ✔ should emit Withdrawn event
            ✔ should transfer the token to the deposit owner

    Integration tests - Depositing
      when no token was deposited yet
        ✔ should have depositCount equal to 0
        ✔ should have no tokens deposited
      when depositing tokens
        ✔ should update depositCount
        ✔ should update token balances
        ✔ should update saved deposits details
      when locking existing deposits
        ✔ should not change depositCount
        ✔ should not change token balances
        ✔ should update saved deposits details
      when depositing tokens with a lock
        ✔ should update depositCount
        ✔ should update token balances
        ✔ should update saved deposits details
      when extending the lock of existing deposits
        ✔ should not change depositCount
        ✔ should not change token balances
        ✔ should update saved deposits details
      when withdrawing deposits
        ✔ should not change depositCount
        ✔ should update token balances
        ✔ should update saved deposits details
      when depositing tokens again
        ✔ should update depositCount
        ✔ should update token balances
        ✔ should update saved deposits details

    Integration tests - Lock Period
      when updating allowed lock period range
        when minimum lock period is increased
          ✔ should allow to use new minimum lock period for new deposits
          ✔ should allow to extend the lock of the existing deposits
        when minimum lock period is decreased
          ✔ should allow to use new minimum lock period for new deposits
          ✔ should not allow to decrease the lock period of the existing deposits
        when maximum lock period is increased
          ✔ should allow to use new maximum lock period for new deposits
          ✔ should allow to extend the lock of the existing deposits
        when maximum lock period is decreased
          ✔ should allow to use new maximum lock period for new deposits
          ✔ should not allow to decrease the lock period of the existing deposits

    Integration tests - Supported Tokens
      when updating supported tokens
        when new token can only be deposited
          ✔ should make a deposit of the new token
          ✔ should not lock the deposit of the new token
          ✔ should withdraw the deposit of the new token
      when new token can be deposited and locked
        ✔ should make a deposit of the new token
        ✔ should allow to lock the deposit of the new token later
        ✔ should make a deposit of the new token with a lock
        ✔ should extend the lock the deposit of the new token
        ✔ should withdraw the deposits of the new token after lock period


  186 passing (4s)

Tests for 'thesis/orangekit'

  BitcoinSafeOwner
    using test harness
      constructor
        ✔ should set initialized property
      setup
        when contract is initialized
          ✔ should revert with ContractAlreadyInitialized
        when contract is not initialized
          when truncatedBitcoinAddress is zero
            ✔ should revert with InvalidTruncatedBitcoinAddress
          when emergencyGovernance address is zero
            ✔ should revert with EmergencyGovernanceAddressZero
          when parameters are valid
            ✔ should set truncatedBitcoinAddress
            ✔ should set emergencyGovernance address
      isValidSignature(bytes,bytes)
        when truncatedBitcoinAddress is not set
          ✔ should revert with InvalidTruncatedBitcoinAddress
      isValidSignature(bytes32,bytes)
        when truncatedBitcoinAddress is not set
          ✔ should revert with InvalidTruncatedBitcoinAddress
      encodeDigest
        ✔ should encode the digest properly (98ms)
      shouldEncodeDigest
        when the highest v bit is not set
          ✔ should not decode (66ms)
        when the highest v bit is set
          ✔ should decode (65ms)
    when contract is deployed by the OrangeKitSafeFactory
      setup
        ✔ should set truncatedBitcoinAddress
        ✔ should set emergencyGovernance address
        when called again
          ✔ should revert with ContractAlreadyInitialized
      isValidSignature(bytes,bytes)
        when signature is valid
          ✔ should return 0x20c13b0b
        when signature is valid (encoded digest mode)
          ✔ should return 0x20c13b0b for vector 1 (89ms)
          ✔ should return 0x20c13b0b for vector 2 (101ms)
          ✔ should return 0x20c13b0b for vector 3 (89ms)
          ✔ should return 0x20c13b0b for vector 4 (95ms)
          ✔ should return 0x20c13b0b for vector 5 (135ms)
        when signature is invalid
          ✔ should return 0xffffffff
        when signature is too short
          ✔ should revert with InvalidSignatureLength
        when signature is too long
          ✔ should revert with InvalidSignatureLength
        when public key is not on the curve
          ✔ should revert with PubkeyNotOnCurve (39ms)
        when s is in the upper range
          ✔ should revert with InvalidSignatureS (40ms)
      isValidSignature(bytes32,bytes)
        when signature is valid
          ✔ should return 0x1626ba7e
        when signature is valid (encoded digest mode)
          ✔ should return 0x1626ba7e for vector 1 (95ms)
          ✔ should return 0x1626ba7e for vector 2 (83ms)
          ✔ should return 0x1626ba7e for vector 3 (92ms)
        ✔ should return 0x1626ba7e for vector 4 (102ms)
        ✔ should return 0x1626ba7e for vector 5 (87ms)
      when signature is invalid
        ✔ should return 0xffffffff
      when signature is too short
        ✔ should revert with InvalidSignatureLength
      when signature is too long
        ✔ should revert with InvalidSignatureLength
  compatibility tests with OrangeKitSafeFactory
    uncompressed P2PKH
      when the v offset is valid
        ✔ should return 0x20c13b0b with offset 0
      when the v offset is incompatible
        ✔ should return 0xffffffff on offset 4
        ✔ should return 0xffffffff on offset 8
        ✔ should return 0xffffffff on offset 12
      when the v offset is unsupported
        ✔ should throw InvalidSignatureV on offset 2
        ✔ should throw InvalidSignatureV on offset 6
        ✔ should throw InvalidSignatureV on offset 10
        ✔ should throw InvalidSignatureV on offset 14
    compressed P2PKH
      when the v offset is valid
        ✔ should return 0x20c13b0b with offset 4
      when the v offset is incompatible
        ✔ should return 0xffffffff on offset 0
        ✔ should return 0xffffffff on offset 8 (49ms)
      when the v offset is unsupported
        ✔ should throw InvalidSignatureV on offset 2 (170ms)
        ✔ should throw InvalidSignatureV on offset 6 (43ms)
        ✔ should throw InvalidSignatureV on offset 10
        ✔ should throw InvalidSignatureV on offset 14
    P2SH_P2WPKH
      when the v offset is valid
        ✔ should return 0x20c13b0b with offset 8
      when the v offset is incompatible
        ✔ should return 0xffffffff on offset 0
        ✔ should return 0xffffffff on offset 4
        ✔ should return 0xffffffff on offset 12
      when the v offset is unsupported
        ✔ should throw InvalidSignatureV on offset 2
        ✔ should throw InvalidSignatureV on offset 6
        ✔ should throw InvalidSignatureV on offset 10
        ✔ should throw InvalidSignatureV on offset 14
    P2WPKH
      when the v offset is valid
        ✔ should return 0x20c13b0b with offset 12
      when the v offset is incompatible
        ✔ should return 0xffffffff on offset 0
        ✔ should return 0xffffffff on offset 8
      when the v offset is unsupported
        ✔ should throw InvalidSignatureV on offset 2
        ✔ should throw InvalidSignatureV on offset 6
        ✔ should throw InvalidSignatureV on offset 10
        ✔ should throw InvalidSignatureV on offset 14
BitcoinSafeOwner - Upgrade
  DOMAIN_SEPARATOR
    ✔ should be keccak256 of EIP712 domain struct
  UPGRADE_SINGLETON_TYPEHASH
    ✔ should be keccak256 of the UpgradeSingleton message
  upgradeSingleton
    when upgrading to zero address
      ✔ should revert
    when upgrading to the same address
      ✔ should revert
    when the upgrade signature is incorrect
      ✔ should revert
    when the upgrade signature is correct
      when init data are empty
        ✔ should revert (49ms)
      when init data are not empty
        ✔ should upgrade the singleton address
```

```
          ✔ should emit SingletonUpgraded event
          ✔ should call the setup function
          ✔ should remain functional
        when init data are not empty and initialization failed
          ✔ should revert (64ms)
        when trying to use the same upgrade signature again
          ✔ should revert
    emergencyUpgradeSingleton
      when called by a third party
        ✔ should revert
      when called by the emergency upgrader
        when upgrading to zero address
          ✔ should revert
        when upgrading to the same address
          ✔ should revert
        when called after the emergency upgrades were disabled
          ✔ should revert
        when called while emergency upgrades are enabled
          when init data are empty
            ✔ should revert
          when init data are not empty
            ✔ should upgrade the singleton address
            ✔ should emit SingletonUpgraded event
            ✔ should call the setup function
            ✔ should remain functional
  EmergencyGovernance
    emergencyUpgrader
      when emergency upgrades are enabled
        ✔ should return the upgrader address
      when emergency upgrades are disabled
        ✔ should revert
    disable
      when called by a third party
        ✔ should revert
      when called by the contract owner
        ✔ should disable emergency upgrades
        ✔ should emit an event
      when called by the contract owner one more time
        ✔ should revert
    setEmergencyUpgrader
      when called by a third party
        ✔ should revert
      when called by the contract owner
        when emergency upgrades are disabled
          ✔ should revert
        when emergency upgrades are enabled
          ✔ should replace the emergency upgrader
          ✔ should emit EmergencyUpgraderChanged event
  OrangeKitDeployer
    deployEmergencyGovernance
      ✔ should set the address and emit event
      ✔ should deploy the EmergencyGovernance contract
    deployBitcoinSafeOwnerSingleton
      ✔ should set the address and emit event
      ✔ should deploy the BitcoinSafeOwner singleton contract
    deployOrangeKitSafeFactorySingleton
      ✔ should set the address and emit event
      ✔ should deploy the OrangeKitSafeFactory singleton contract
    deployOrangeKitSafeFactoryProxy
      when EmergencyGovernance is not deployed
        ✔ should revert
      when BitcoinSafeOwner singleton is not deployed
        ✔ should revert
      when OrangeKitSafeFactory singleton is not deployed
        ✔ should revert
      when all other contracts are deployed
        ✔ should set the address and emit event
        ✔ should deploy the OrangeKitSafeFactory proxy contract
        ✔ should initialize the deployed OrangeKitSafeFactory proxy contract
        ✔ should transfer the ownership of the OrangeKitSafeFactory proxy contract
    deploy
      ✔ should set all addresses and emit events
      ✔ should deploy all contracts (43ms)
      ✔ should initialize the deployed OrangeKitSafeFactory proxy contract
      ✔ should transfer the ownership of the OrangeKitSafeFactory proxy contract
  OrangeKitSafeFactory
    initialize
      when called on initialized contract
        ✔ should revert
      when called on uninitialized contract
        when safe singleton is zero address
          ✔ should revert
        when safe owner singleton is zero address
          ✔ should revert
        when emergency governance is zero address
          ✔ should revert
        when safe singleton is EOA
          ✔ should revert
        when safe owner singleton is EOA
          ✔ should revert
        when emergency governance is EOA
          ✔ should revert
        when called with correct parameters
          ✔ should deploy contract and set parameters correctly
    deploySafe
      when called with a zero address bitcoin owner
        ✔ should revert
      when called for the same owner more than once
        ✔ should revert
      when called once
        ✔ should set BitcoinSafeOwner as the only BitcoinSafe owner
        ✔ should set bitcoin signer ethereum address in the BitcoinSafeOwner contract
        ✔ should set emergency governance in the BitcoinSafeOwner contract
        ✔ should emit the SafeDeployed event
      when called for the same owner at different order
        ✔ should yield the same addresses (1486ms)
    predictAddresses
      for one bitcoin signer
        ✔ should predict correct addresses (42ms)
      for multiple bitcoin signers
        ✔ should predict correct addresses for bitcoin signer 1
        ✔ should predict correct addresses for bitcoin signer 2
        ✔ should predict different addresses for two signers
    transferOwnership
      when called by a third party
        ✔ should revert
      when called by the owner
        when called with zero address new owner
          ✔ should revert
        when called with non-zero new owner address
          ✔ should transfer the ownership
          ✔ should emit OwnershipTransferred event
    upgradeSingleton
      when called by a third party
        ✔ should revert
      when called by the owner
        when upgrading to zero address
          ✔ should revert
        when upgrading to the same address
          ✔ should revert
        when init data are empty
          ✔ should revert
        when init data are not empty
          ✔ should upgrade the singleton address
          ✔ should emit SingletonUpgraded event
          ✔ should call the initialize function
          ✔ should remain functional
  OrangeKitSafeFactory - contract upgrades
    when upgrading to a valid contract
      when singletons remain the same
        contract variables
          ✔ should initialize new variable
          ✔ should keep old singletons addresses
        predictAddresses
        ✔ should predict correct addresses for bitcoin signer 1
        ✔ should predict correct addresses for bitcoin signer 2
      deploySafe
        for safe that has not been deployed in V1
          ✔ should deploy with addresses predicted in V1
        for safe that was already deployed in V1
          ✔ should revert
  Safe with Bitcoin Owner
    deploy safe and test transaction signing
      uncompressed P2PKH address
        safe deployment
          ✔ should set BitcoinSafeOwner as the only BitcoinSafe owner
          ✔ should set truncatedBitcoinAddress in the BitcoinSafeOwner contract
        token transfer execution in the safe
          ✔ should emit ExecutionSuccess event
          ✔ should transfer tokens from safe to destination
      compressed P2PKH address
        safe deployment
          ✔ should set BitcoinSafeOwner as the only BitcoinSafe owner
          ✔ should set truncatedBitcoinAddress in the BitcoinSafeOwner contract
        token transfer execution in the safe
          ✔ should emit ExecutionSuccess event
          ✔ should transfer tokens from safe to destination
      P2SH.P2WPKH address
        safe deployment
          ✔ should set BitcoinSafeOwner as the only BitcoinSafe owner
          ✔ should set truncatedBitcoinAddress in the BitcoinSafeOwner contract
        token transfer execution in the safe
          ✔ should emit ExecutionSuccess event
          ✔ should transfer tokens from safe to destination
      P2WPKH address
        safe deployment
          ✔ should set BitcoinSafeOwner as the only BitcoinSafe owner
          ✔ should set truncatedBitcoinAddress in the BitcoinSafeOwner contract
        token transfer execution in the safe
          ✔ should emit ExecutionSuccess event
          ✔ should transfer tokens from safe to destination
  bitcoinSafeOwner helpers
    recoverTruncatedBitcoinAddressFromBase58
      ✔ should recover the correct data from a uncompressed P2PKH address
      ✔ should recover the correct data from a compressed P2PKH address
      ✔ should recover the correct data from a P2SH.P2WPKH address
    recoverTruncatedBitcoinAddressFromBech32
      ✔ should recover the correct data from a P2WPKH address
  169 passing (13s)
```

Code Coverage

Update: The coverage situation remains largely the same.

Coverage appears to be decent for code in scope in `keep-network/tbtc-v2` and `thesis/mezo-portal`. However, it looks like there are two `revert` statements that are not being tested at the following locations:

1. `Portal.sol#L162`
2. `BitcoinDepositor.sol#L198`

Coverage for `keep-network/tbtc-v2`

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	0	0	0	0	
GovernanceUtils.sol	0	0	0	0	... 36,37,38,40
contracts/bank/	97.87	93.75	100	98.33	
Bank.sol	97.87	93.75	100	98.33	380

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
IReceiveBalanceApproval.sol	100	100	100	100	
contracts/bridge/	2.83	1.75	1.21	2.24	
BitcoinTx.sol	54.05	27.78	40	55	... 352,371,373
Bridge.sol	0	0	0	0	... 1,1966,1991
BridgeGovernance.sol	0	0	0	0	... 2,1767,1783
BridgeGovernanceParameters.sol	0	0	0	0	... 9,1571,1572
BridgeState.sol	0	0	0	0	... 852,857,858
Deposit.sol	0	0	0	0	... 420,427,435
DepositSweep.sol	0	0	0	0	... 569,572,574
EcdsaLib.sol	100	100	100	100	
Fraud.sol	0	0	0	0	... 576,577,578
Heartbeat.sol	100	100	100	100	
IRelay.sol	100	100	100	100	
MovingFunds.sol	0	0	0	0	... 8,1069,1072
Redemption.sol	0	0	0	0	... 6,1191,1193
RedemptionWatchtower.sol	0	0	0	0	... 618,620,621
VendingMachine.sol	0	0	0	0	... 309,310,311
VendingMachineV2.sol	0	0	0	0	... 109,110,112
VendingMachineV3.sol	0	0	0	0	... 129,130,132
WalletProposalValidator.sol	0	0	0	0	... 877,893,898
Wallets.sol	0	0	0	0	... 706,717,720
contracts/hardhat-dependency-**compiler/**@keep-**network/ecdsa/contracts/**	100	100	100	100	
WalletRegistry.sol	100	100	100	100	
contracts/hardhat-dependency-**compiler/@openzeppelin/contracts/proxy/transparent/**	100	100	100	100	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
ProxyAdmin.sol	100	100	100	100	
TransparentUpgradeableProxy.sol	100	100	100	100	
contracts/integrator/	100	87.5	100	100	
AbstractTBTCDepositor.sol	100	87.5	100	100	
IBridge.sol	100	100	100	100	
ITBTCVault.sol	100	100	100	100	
contracts/l2/	50.38	46.53	59.09	48.6	
L1BitcoinDepositor.sol	0	0	0	0	... 634,636,651
L2BitcoinDepositor.sol	0	0	0	0	... 175,181,186
L2TBTC.sol	100	97.62	100	100	
L2WormholeGateway.sol	100	81.25	100	100	
Wormhole.sol	100	100	100	100	
contracts/maintainer/	0	0	0	0	
MaintainerProxy.sol	0	0	0	0	... 536,553,558
contracts/relay/	82.93	68.37	66.67	80.31	
LightRelay.sol	100	90.54	100	98.08	438,439
LightRelayMaintainerProxy.sol	0	0	0	0	... 138,140,142
contracts/test/	65	33.33	56.9	56.36	
BankStub.sol	100	100	0	0	9
BridgeStub.sol	0	0	0	0	... 158,166,172
HeartbeatStub.sol	100	100	100	100	
LightRelayStub.sol	100	100	100	100	
ReceiveApprovalStub.sol	0	0	0	0	23,24,27,31
SepoliaLightRelay.sol	0	0	0	0	41,45,46
SystemTestRelay.sol	75	100	50	50	18,22,38,42
TestBitcoinTx.sol	100	100	100	100	
TestERC20.sol	100	100	100	100	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
TestERC721.sol	100	100	100	100	
TestEcdsaLib.sol	100	100	100	100	
TestTBTCDepositor.sol	95.65	57.14	88.24	95.24	163,225
WormholeBridgeStub.sol	90	100	87.5	92.31	123
contracts/token/	100	100	100	100	
TBTC.sol	100	100	100	100	
contracts/vault/	24.17	15	24.44	22.09	
DonationVault.sol	100	100	100	100	
IVault.sol	100	100	100	100	
TBTCOptimisticMinting.sol	6.35	2.27	9.09	5.38	... 560,561,562
TBTCVault.sol	23.81	13.16	22.22	25	... 343,344,345
All files	19.52	16.64	21.12	18.44	

Coverage for `thesis/mezo-portal`

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	100	93.59	100	98.23	
BitcoinDepositor.sol	100	92.86	100	95.45	198
Portal.sol	100	93.75	100	98.9	162
contracts/interfaces/	100	100	100	100	
IApproveAndCall.sol	100	100	100	100	
IReceiveApproval.sol	100	100	100	100	
contracts/tests/	90.91	50	100	92.31	
MockERC20.sol	83.33	50	100	83.33	36
MockTBTC.sol	100	50	100	100	
contracts/tests/upgrades/	10.53	3.85	13.95	8.33	
PortalV2.sol	31.03	10.61	40	24.72	... 498,506,516
PortalV2MisplacedSlot.sol	0	0	0	0	... 482,490,500
PortalV2MissingSlot.sol	0	0	0	0	... 478,486,496

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
All files	39.61	31.2	43.94	37.18	

Coverage for `thesis/orangekit`

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	99.18	91.96	97.92	96.37	
BitcoinSafeOwner.sol	100	93.1	100	96.7	172,190,219
ERC1271.sol	100	100	100	100	
EmergencyGovernance.sol	100	92.86	100	91.67	64
LegacyERC1271.sol	100	100	100	100	
OrangeKitDeployer.sol	100	87.5	100	96	164
OrangeKitSafeFactory.sol	96.88	93.33	93.33	96.77	100,175
Proxy.sol	100	50	100	100	
contracts/test/	27	12.5	45	31.54	
BitcoinSafeOwnerHarness.sol	100	100	100	100	
BitcoinSafeOwnerV2.sol	7.58	8.93	16.67	12.36	... 574,576,591
OrangeKitSafeFactoryV2.sol	58.62	20.83	56.25	56.36	... 407,409,412
TestERC20.sol	100	100	100	100	
All files	66.67	58.85	73.86	68.13	

Changelog

- 2024-05-03 - Initial report
- 2024-05-24 - Final report

About Quantstamp

Quantstamp is a global leader in blockchain security. Founded in 2017, Quantstamp's mission is to securely onboard the next billion users to Web3 through its best-in-class Web3 security products and services.

Quantstamp's team consists of cybersecurity experts hailing from globally recognized organizations including Microsoft, AWS, BMW, Meta, and the Ethereum Foundation. Quantstamp engineers hold PhDs or advanced computer science degrees, with decades of combined experience in formal verification, static analysis, blockchain audits, penetration testing, and original leading-edge research.

To date, Quantstamp has performed more than 500 audits and secured over $200 billion in digital asset risk from hackers. Quantstamp has worked with a diverse range of customers, including startups, category leaders and financial institutions. Brands that Quantstamp has worked with include Ethereum 2.0, Binance, Visa, PayPal, Polygon, Avalanche, Curve, Solana, Compound, Lido, MakerDAO, Arbitrum, OpenSea and the World Economic Forum.

Quantstamp's collaborations and partnerships showcase our commitment to world-class research, development and security. We're honored to work with some of the top names in the industry and proud to secure the future of web3.

Notable Collaborations & Customers:

- Blockchains: Ethereum 2.0, Near, Flow, Avalanche, Solana, Cardano, Binance Smart Chain, Hedera Hashgraph, Tezos
- DeFi: Curve, Compound, Maker, Lido, Polygon, Arbitrum, SushiSwap
- NFT: OpenSea, Parallel, Dapper Labs, Decentraland, Sandbox, Axie Infinity, Illuvium, NBA Top Shot, Zora
- Academic institutions: National University of Singapore, MIT

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by Quantstamp; however, Quantstamp does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication or other making available of the report to you by Quantstamp.

Notice of confidentiality

This report, including the content, data, and underlying methodologies, are subject to the confidentiality and feedback provisions in your agreement with Quantstamp. These materials are not to be disclosed, extracted, copied, or distributed except to the extent expressly authorized by Quantstamp.

Links to other websites

You may, through hypertext or other computer links, gain access to web sites operated by persons other than Quantstamp. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites&aspo; owners. You agree that Quantstamp are not responsible for the content or operation of such web sites, and that Quantstamp shall have no liability to you or any other person or entity for the use of third-party web sites. Except as described below, a hyperlink from this web site to another web site does not imply or mean that Quantstamp endorses the content on that web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the report. Quantstamp assumes no responsibility for the use of third-party software on any website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any output generated by such software.

Disclaimer

The review and this report are provided on an as-is, where-is, and as-available basis. To the fullest extent permitted by law, Quantstamp disclaims all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. You agree that your access and/or use of the report and other results of the review, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE. This report is based on the scope of materials and documentation provided for a limited review at the time provided. You acknowledge that Blockchain technology remains under development and is subject to unknown risks and flaws and, as such, the report may not be complete or inclusive of all vulnerabilities. The review is limited to the materials identified in the report and does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. The report does not indicate the endorsement by Quantstamp of any particular project or team, nor guarantee its security, and may not be represented as such. No third party is entitled to rely on the report in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. Quantstamp does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party, or any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, or any related services and products, any hyperlinked websites, or any other websites or mobile applications, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third party. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

Quantstamp